Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления

Яушева Ольга Александровна

Магистерская диссертация

Автоматическое определение событий международного значения на основе анализа новостных лент на различных языках

Направление 01.04.02
«Прикладная математика и информатика»
Магистерская программа «Прикладные информационные технологии. Информационные экспертные системы»

Научный руководитель,
ст.преп.,
Малинина М.А

Санкт-Петербург
2018

Содержание

Введение

В связи со стремительным развитием сети Интернет объемы информации, получаемые конечными пользователями, значительно возрастают. Развитие сети повлекло за собой увеличение количества новостных интернет–агентств, которые заменили традиционные средства информации. Однако, из-за отсутствия структуры и неоднородности источников информации доступ к этому огромному набору текстов ограничивается просмотром, поиском и чтением статей. Большое число источников информации, а также их объем, привели к необходимости быстрого анализа и обработки новостного потока. Одной из задач, которая требует решения, является предоставление пользователю информации о конкретном событии или же личности из различных источников новостей. Для того, чтобы конечный пользователь имел возможность сформировать наиболее полное представление о происходящем событии с учетом различных точек зрения.

Одним из решений требуемой задачи является интеллектуальный анализ текста (англ. *text mining, TM*) [1]. Его целью является получение разного рода информации из набора текстовых документов с помощью методов машинного обучения (англ. *machine learning, ML*). К типичным задачам анализа текста относятся: категоризация текстовых документов, кластеризация текстовых документов, извлечение информации из текста, информационный поиск.

Извлечение информации (англ. *information extraction, IE*) определяется как процесс выборочного структурирования и объединения данных, которые указаны явно в одном или нескольких документах на естественном языке (англ. *natural language*). Другими словами, задача IE заключается в получении структурированной фактической информации из неструктурированной.

Большое количество знаний и информации, хранится в текстовых документах. Чаще всего эти тексты доступны только в неструктурированном представлении, так как они создаются и интерпретируются людьми. Для того, чтобы воспользоваться преимуществами этого огромного количества скрытой информации, и включить ее в аналитические процессы, необходимо преобразовать информацию в структурированное представление. IE решает именно эту задачу. Система IE пытается идентифицировать четко определенные объекты и отношения в неструктурированных данных, особенно в текстовых документах.

Помимо извлечения и идентификации имен и сущностей, извлечение

отношений и событий является основной задачей IE. События – это реальные процессы, которые разворачиваются во времени и пространстве и, как правило, связаны с определенным изменением состояния. Формально задача извлечения событий состоит в том, чтобы автоматически идентифицировать события в свободном тексте и получить подробную информацию о них. Идеальная система извлечения событий отвечает на вопрос: кто и что сделал? Из-за сложности и неоднозначности естественного языка извлечение событий является нетривиальной задачей.

Постановка задачи

Пусть пользователь имеет новостную ленту – некоторый набор сообщений о различных событиях, происходящих в мире. Читатель, ознакомившись с новостным сообщением, хочет узнать альтернативное мнение о событии, которое представлено в тексте. Одно и то же событие может быть по разному описано в различных источниках информации, особенно это касается популярных и масштабных событий. В случае, когда событие является значимым в мировом масштабе, пользователю выгодно иметь представление о том, что пишут об этом событии СМИ других стран.

Целью данной работы является построение автоматизированной системы определения международного события и предоставления конечному пользователю информации об уровне значимости события. Данная информация собирается на основе предложенной пользователем новости, с использованием альтернативных новостных порталов на различных языках.

В качестве новостной ленты пользователя будем рассматривать статьи русскоязычных новостных интернет–агентств. В качестве альтернативных источников информации – иностранные новостные порталы на английском, немецком и французском языках.

Событие определяется как набор атрибутов – ключевых слов или словосочетаний, которые отвечают на вопрос: кто и что сделал? Событие будет считаться международным, если имеются статьи о данном событии в альтернативных источниках информации.

Задачи, которые должна решать автоматизированная система:

1. Сбор исходных данных – новостных статей.
2. Обработка новостных документов.
3. Определение и извлечение событий из новости.
4. Проверка события, для определения является ли событие международным.
5. Предоставление пользователю новости об идентифицированном событии из альтернативных источников.

Глава 1. Задача извлечения информации

1.1. Описание предметной области

Information Extraction (IE) – задача извлечения информации – рассматривает вопросы идентификации определенных сущностей и отношений в неструктурированных данных, особенно в текстовых документах [2]. Таким образом, извлечение информации из текста становится ключевым компонентом в процессе интеграции текстовых данных, и может рассматриваться в качестве обобщающего термина для многих интересных задач, таких как распознавание имен сущностей, анализ тональности текста или извлечение знаний.

Извлечение событий из неструктурированных данных, таких как новостные сообщения, может быть полезно для дальнейшего практического применения. Например, если система IE в состоянии определить событие, то это может повысить производительность персонализированных информационных систем, так как новостное сообщение может быть выбрано более точно, в зависимости от предпочтений пользователя.

Извлечение информации является широким полем исследований и тесно связанно с несколькими дисциплинами [3]:

- Natural Language Processing (NLP) – обработка естественного языка.
- Text Mining (TM) – интеллектуальный анализ текста.
- Machine Learning (ML) – машинное обучение.

Задача извлечения информации не получила такого широкого внимания, как информационный поиск (англ. *Information Retrieval, IR*), и часто смешивается с последним [3]. Задача информационного поиска текста состоит в том, чтобы выбрать из набора текстовых документов подмножество, которое имеет отношение к конкретному запросу, на основе поиска по ключевым словам. Процесс IR обычно возвращает ранжированный список

документов, где ранг соответствует баллу релевантности, который система присвоила документу в ответ на запрос. Однако ранжированный список документов не предоставляет подробную информацию о содержании этих документов. Цель IE не ранжировать или выбрать документы, а извлечь из документов важные характеристики о предварительно определенных типах событий, сущностях или отношениях. Подводя итог, IE стремится преобразовывать коллекции текстовых данных в форму, которая облегчает поиск и обнаружение знаний в таких коллекциях.

Системы IE в целом более трудны и наукоемки для построения, чем системы IR. Однако методы IE и IR могут рассматриваться как взаимодополняющие и потенциально могут быть объединены различными способами. IR часто используется в IE для предварительной фильтрации очень большой коллекции документов и сведении её к управляемому подмножеству, к которому могут быть применены методы IE. В качестве альтернативы, IE может использоваться как субкомпонента IR-системы для идентификации структур или для интеллектуального индексирования документов.

Рассмотрим в качестве примера извлечение информации о событии из предложения, мы заинтересованы в определении времени проведения, выявлении основных участников этого мероприятия и его местонахождении:

«В июне в Санкт-Петербурге прошел фестиваль цветов, в котором приняли участие школьники города.»

В таблице 1.1 пример структурированной информации, которая получена из указанного выше предложения. Процесс извлечения такой структурированной информации включает в себя идентификацию некоторых мелких структур, таких как словосочетания, обозначающие лицо или группу лиц, географические ссылки и нахождение семантических отношений между ними.

Таблица 1.1: Пример извлечения информации из предложения.

Время проведения	июнь
Место проведения	Санкт-Петербург
Участники	школьники

1.2. Основные определения IE

В приложении 1 представлен список обозначений и сокращений терминов, которые используются в данной работе.

Задача извлечения информации заключается в выявлении заранее заданного класса сущностей, отношений и событий в текстах на естественном языке, а также извлечения соответствующих свойств (аргументов) идентифицированных сущностей, отношений или событий.

Извлекаемая информация предварительно задается в пользовательских структурах, называемых **шаблонами**, каждый из которых состоит из нескольких **слотов** (или **атрибутов**) которые должны быть созданы системой IE при обработке текста. Обычно слот заполняют: строки из текста, одно из нескольких предопределенных значений или ссылка на ранее созданный шаблон объекта. Система IE создает структурированное представление (например, записи базы данных) выбранной информации, извлеченной из анализируемого текста.

Рассмотрим пример:

«В мае Ивана Ивановича назначили учителем математики средней общеобразовательной школы.»

В данном случае рассмотрим событие – назначение. Шаблон может быть представлен следующим образом:

<Назначение[Лицо; Должность; Дата]>

В представленном примере шаблон <Назначение> содержит три слота (или атрибута). Первый слот – лицо, описывает сущность назначаемого, второй слот – должность, описывает на какую должность назначили сущность, третий слот – дата, описывает дату (или время) назначения. Пример структурирования извлеченной информации представлен в таблице 1.2.

Таблица 1.2: Пример извлечения информации из текста.

Назначение	
Лицо	Иван Иванович
Должность	учитель математики
Дата	май

1.3. Используемые методы извлечения информации

Применение извлечения информации к тексту связано с задачей упрощения текста с целью создания структурированного представления информации, присутствующей в свободном тексте. Общая цель состоит в том, чтобы создать более удобный машиносчитываемый текст для обработки предложений.

К типичным задачам извлечения информации относятся [4]:

- Named entity recognition (NER) – распознавание именованных сущностей.
- Coreference resolution (CO) – разрешение кореференции.
- Relationship extraction – извлечение отношений.

1.3.1. Named entity recognition

Named entity recognition (NER) – распознавание именованных сущностей.

NER решает проблему идентификации (обнаружения) предопределенных типов именованных сущностей, таких как:

- Организации (например, «Санкт-Петербургский государственный университет», «Министерство внутренних дел»).
- Лица (например, «Дмитрий Медведев», «Петросян Леон Аганесович»).
- Временные выражения (например «9:20 a.m.», «1 января 2017»).
- Географические названия (например, «Финский залив», «Петергоф»).
- Числовые и валютные выражения (например, «10 тысяч рублей»).

и так далее.

Задача NER может дополнительно включать в себя извлечение описательной информации из текста путем заполнения шаблонов. Например, в случае лиц, она может включать в себя извлечение звания, должности, национальности, пола и прочих атрибутов человека.

1.3.2. Coreference resolution

Coreference resolution (CO) – разрешение кореференции.

Задача CO рассматривает идентификацию нескольких различных упоминаний одного и того же объекта в тексте. Примерами кореференции могут быть [5]:

- Анафора, например в предложении «Мама мыла посуду, она устала» слова <мама> и <она> – это различные упоминания одного и того же объекта реального мира.
- Синонимия, например «В.В. Путин», «Путин» и «Президент РФ».

Синонимия может быть выражена различными способами:

- Аббревиатура, например «факультет прикладной математики – процессов управления» и «ПМ – ПУ».
- Транслитерация, например «Google» и «Гугл».
- Синоним, «компания» и «организация».

Разрешение кореференции является важным компонентом IE системы и помогает избежать ложных извлечений из текста ненужных объектов и фактов.

1.3.3. Relationship extraction

Relationship extraction – извлечение отношений.

Извлечение отношений из текста является задачей обнаружения предопределенных отношений между сущностями, которые указаны в тексте. Примерами могут служить:

- Отношение <Местоположение[Кто; Где]> из предложения «Иван находится в Санкт-Петербурге» заполняется как <Местоположение[Иван; Санкт-Петербург]>.
- Отношение <Материнство[Ребёнок; Мать]> из предложения «Светлана Ивановна является мамой Пети» заполняется как <Материнство[Петя; Светлана Ивановна]>.

1.4. Подходы к извлечению информации

Подходы к извлечению информации можно условно разделить на три основные категории [4] [5]:

- основанные на правилах (англ. *rule-based*).

- на основе онтологий (англ. *ontology-based*).
- основанные на машинном обучении.

Также возможны варианты комбинирования данных подходов, с учетом их достоинств и недостатков. Такие системы называются гибридными.

Подход основанный на правилах требует участия эксперта, который составляет шаблоны для извлечения информации. В зависимости от предметной области составляется описание правил, которые необходимы для извлечения фактов или объектов. Написанный шаблон применяется к документу. Выявляются цепочки текста, которые соответствуют правилу.

В качестве примера можно рассмотреть извлечение названий улиц из текста, для этого необходимо составить правило: если встречаются слова <ул>, <ул.>, а также все словоформы лексемы <улица> (<улице>, <улицу> и так далее), а за ними следует слово с прописной буквы, то извлекаем найденное слово и помечаем его как название улицы. Таким образом, применив это правило к предложениям «Сегодня состоится собрание на ул. Ленина», «Сегодня на улице холодно» получим один извлеченный объект – <ул. Ленина>.

Данный подход широко применяется и хорошо справляется с извлечением имён, дат, временных промежутков, а также числовых и денежных выражений. Достоинством данного метода является гибкость. Эксперт может изменить и дополнить шаблон, если правило недостаточно полно охватывает необходимые для извлечения объекты. К недостаткам можно отнести сложность написания правил.

Второй подход основан на онтологиях [6]. Онтология – форма представления знаний, в которой содержатся различные понятия и отношения между ними, а также их характеристики. Обычно онтология задается для конкретной области. Например, геополитическая онтология, которая определяет такие понятия, как страна, провинция и город. Примером онтологии может служить Википедия [7]. При использовании подхода на основе онтологий, мы можем предполагать и строить гипотезы, чтобы в дальнейшем принимать или отклонять их. Пример онтологии представлен на рисунке 1.

В качестве примера рассмотрим фразу «Президент России Владимир Путин». При отправлении ключевых слов в онтологию – Википедия мы получим набор ссылок на статьи, такие как <Президент>, <Президент Российской Федерации>, <Россия>, <Владимир>, <Владимир Путин>.

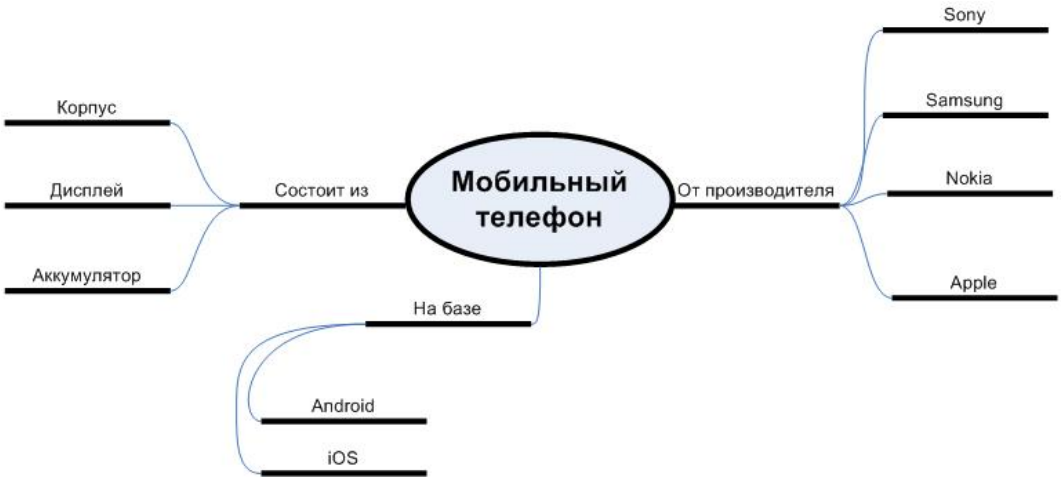

Рис. 1: Пример визуализации онтологии для предметной области «мобильный телефон».

Таким образом, мы можем извлечь из данного предложения именованную сущность или должность.

Достоинствами данного метода являются высокая точность извлечения именованных сущностей. К недостаткам можно отнести низкую полноту извлечения, которая зависит от размера онтологии.

Третий подход основан на машинном обучении. Для реализации этого подхода, необходим размеченный корпус документов, в котором для каждого слова необходимо разметить морфологию, синтаксис, а также связи между словами. Далее на основе размеченного корпуса система строит предположения, когда и в каком виде появится некая лексическая единица. К достоинствам данного метода относится отсутствие надобности создания правил или онтологий. К недостаткам же можно отнести разметку обучающего корпуса, так как это очень трудоемкая задача.

1.5. Архитектура системы IE

Типичная система IE включает в себя несколько этапов [8]:

Предварительная обработка входных текстов. Текстовые корпуса часто состоят из неструктурированных, «сырых» текстов на естественном языке. Большую часть релевантной информации можно выделить с помощью закономерностей, которые обнаружены в лингвистических свойствах текстов. Таким образом, благодаря лингвистическому анализу, можно определить важные особенности текста. Для извлечения информации используются следующие лингвистические компоненты:

- **Токенизация**. Текст – это последовательность символов. Цель токе-

низации состоит в том, чтобы определить элементарные части естественного языка, такие как: слова, знаки препинания – такие элементы называются токенами. Полученная последовательность значимых токенов является основой для дальнейшей лингвистической и любой текстовой обработки.

- **Разделение предложений**. Предложения являются одним из наиболее важных элементов естественного языка для структурированного представления письменного содержания. Предложения являются наименьшими единицами для выражения завершенных мыслей или событий. Поэтому правильное определение границ предложения имеет важное значение для многих подходов IE. Данная задача была бы тривиальной, если бы знаки препинания не использовались неоднозначно. Для синтаксического анализа необходимо правильное представление текста в виде последовательности предложений.

- **Морфологический анализ**. Некоторые факты обычно выражаются определенными частями речи (например, имена - существительными). Определение частей речи токенов называется POS-тегированием (от англ. *part-of-speech tagging* – морфологическая разметка). Системы на основе машинного обучения могут использовать POS-теги в качестве классификационных признаков, системы на основе правил – в качестве элементов правил извлечения.

- **NER, CO**. Задачи распознавания именованных сущностей и разрешения кореференции именованных сущностей являются одними из наиболее часто решаемых системами IE. Некоторые подходы используют простой поиск в предопределенных списках (например, названия городов, стран), некоторые используют обучаемые скрытые Марковские модели для идентификации именованных сущностей и их типа. Задача CO состоит в нахождении нескольких ссылок на один и тот же объект в тексте. Это особенно важно, поскольку соответствующее содержание может быть выражено местоимениями и обозначениями (например, «она устала»). Обе задачи требуют глубокого семантического анализа и не так надежны, как другие лингвистические компоненты.

Для систем, основанных на онтологиях и правилах, лингвистическая предварительная обработка является одним из базовых элементов. Для систем, основанных на машинном обучении предварительная обработка текстов является необязательной, но может оказать серьезное влияние на качество извлечения.

Применение модели извлечения информации. Диапазон при-

менения современных систем IE должен быть максимально широким. Особенности конкретной области не могут быть закреплены в системе, поскольку усилия по адаптации к другим областям слишком велики. Современные системы используют компонент обучения для уменьшения зависимости от конкретных областей и уменьшения объема ресурсов, предоставляемых человеком. Модель извлечения определяется в соответствии с применяемым подходом, и ее параметры оптимизируются с помощью процедуры обучения. Подходы на основе машинного обучения изучают, например, соответствующие классификационные признаки, вероятности. Подходы основанные на правилах изучают набор правил извлечения. Подходы на основе онтологий изучают структуры для дополнения и интерпретации своих знаний для последующего извлечения. Задача состоит в том, чтобы найти модель извлечения, которая позволяет изучить все соответствующие параметры области.

Учитывая проблемы и сложность IE, контролируемое обучение представляется наиболее подходящим и является широко используемой техникой обучения. Большинство подходов предпочитают аннотированные учебные корпуса, хотя некоторые полагаются на человеческий надзор на этапе обучения. Для оценки качества подхода учебный текстовый корпус создается путем аннотирования фрагментов текстов с релевантным содержанием, разделенных на две части. Одна часть, обучающий набор, используется для обучения (изучение параметров модели извлечения), а другая, тестовый набор, используется для проверки способности модели правильно извлекать новую информацию, на которой она не была обучена. Результаты теста также могут быть использованы для улучшения работы модели извлечения.

Некоторые подходы позволяют дальнейшее усовершенствование модели извлечения на основе обратной связи от человека об извлечениях во время применения. Новые оцененные извлечения могут быть включены как новые обучающие экземпляры, и модель может быть переобучена. Компонент обучения имеет решающее значение для системы IE, поскольку он включает в себя алгоритмы идентификации соответствующих частей текста и передачи их в соответствии с целевой структурой.

Постобработка выводных текстов. Основной мотивацией для IE является структурированное представление информации, позволяющее выполнять формальные запросы и автоматическую обработку. Одной из возможностей структурирования извлеченных данных является моделирование целевой структуры как отношения базы данных. После того, как соответствующая информация была найдена с помощью применения моде-

ли извлечения, идентифицированным фрагментам текста присваиваются соответствующие атрибуты целевой структуры. Они могут быть нормализованы в соответствии с ожидаемым форматом (например, представление дат и чисел). Некоторые идентифицированные факты могут появляться в тексте более одного раза или уже находиться в базе данных. В этом случае различные экземпляры могут быть объединены. Наконец, идентифицированная, нормализованная и унифицированная информация хранится в соответствующей базе данных.

Визуализация архитектуры типичной системы извлечения информации из текста представлена на рис. 2.

1.6. Оценка эффективности

Принимая во вниманиея входной текст или набор документов, ожидаемый вывод системы IE может быть определен очень точно. Это облегчает оценку различных систем и подходов к IE. Для этих целей исследовательским сообществом были приняты метрики точности (англ. *precision*) и полноты (англ. *recall*) [3] [4]. Они измеряют эффективность системы с точки зрения пользователя. Определим их формально:

Пусть K обозначает общее количество слотов ответа системы, которые должны быть заполнены в соответствии с аннотированным справочным корпусом (то есть должны быть заполнены по мнению эксперта). Данный корпус представляет собой «истину» или «золотой стандарт».

Пусть C – количество правильно заполненных слотов в ответе системы.

Пусть I – количество неправильно заполненных слотов в ответе системы.

Считается, что слот заполнен неправильно, если он не соответствует слоту в золотом стандарте или если ему присвоено недопустимое значение. Слот считается заполненным правильно, если он соответствует слоту в золотом стандарте.

Точность (P) и полнота (R) могут быть определены следующим образом:

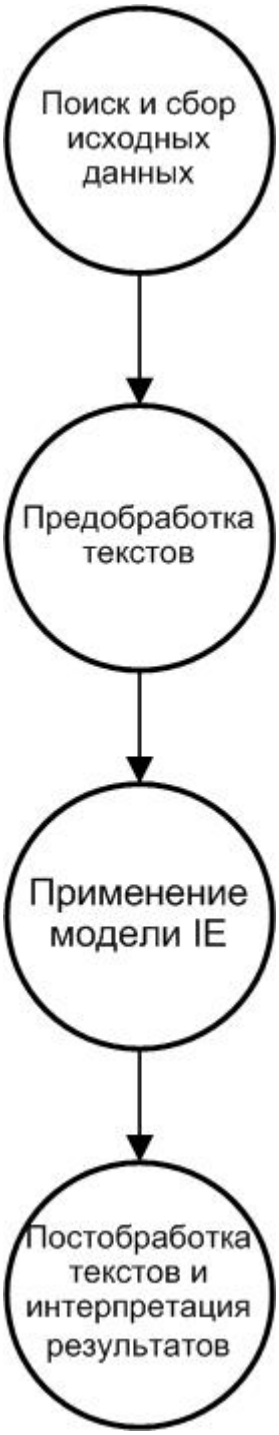

Рис. 2: Архитектура типичной IE системы.

$$P = \frac{C}{C + I} \ , \ R = \frac{C}{K} \tag{1.1}$$

Таким образом, точность (P) – это количество правильных ответов, деленное на количество всех найденных ответов, а полнота (R) – это количество правильных ответов, деленное на общее число правильных ответов «золотого стандарта».

В качестве дополнительной оценки эффективности и качества системы IE служит F-мера. F-мера это соотношение между P и R, которое определяется как гармоническое среднее следующим образом:

$$F = \frac{(\beta^2 + 1) \cdot P \cdot R}{(\beta^2 \cdot P) + R} \tag{1.2}$$

В приведенной выше формуле (1.2) β – есть неотрицательные значение, используемое для корректировки и регулирования «веса» P и R (обычно $\beta = 1$, что дает равный вес для полноты и точности, более низкие значения β увеличивают вес точности).

1.7. Выводы

В настоящей главе было приведено описание предметной области. Введены основные определения задачи IE, используемые методы и подходы к IE, а также архитектура типичной системы.

Задача извлечения информации заключается в получении структурированной информации из неструктурированной. Область IE находится на стыке дисциплин: обработки естественного языка, интеллектуального анализа текста и машинного обучения.

К типичным задачам IE относятся: распознавание именованных сущностей (NER), разрешение кореференции (CO), извлечение отношений (RE). Задача NER заключается в идентификации и извлечении имен сущностей из текста, например: названия организаций, имена лиц, временные выражения, географические названия, числовые и валютные выражения. Задача CO состоит в определении различных упоминаний одного и того же объекта реального мира. Примерами CO могут служить анафора и синонимия. Задача RE – обнаружение отношений между сущностями, которые указаны в тексте.

Подходы к извлечению информации условно делят на три категории: на основе онтологий, на основе правил, на основе машинного обучения. Архитектура типичной системы IE включает в себя несколько этапов: поиск и сбор исходных данных, предварительная обработка текста, модуль извлечения информации, постобработка текста и интерпретация результатов. Для оценки эффективности системы используются оценки полноты, точности и F-меры.

Глава 2. Обзор существующих решений

В данной главе приводится историческая справка о развитии задачи IE. В том числе, рассматриваются существующие решения на основе различных подходов.

2.1. Развитие задачи IE

Важную роль в успешном развитии области извлечения информации сыграли «Конференции по Пониманию сообщений» (англ. *Message Understanding Conferences, MUC*) [9] инициированные ВМС США и спонсируемые агентством управления перспективных исследовательских проектов министерства обороны США (англ *Defense Advanced Research Projects Agency, DARPA*). Конференции были проведены с целью координации нескольких исследовательских групп и правительственных учреждений США, стремящихся улучшить технологии IE и IR [2] [10]. MUC определил несколько типов задач IE и предложил научному сообществу построить свои системы для выполнения этих задач. Участникам MUC первоначально было дано подробное описание задачи IE, наряду с аннотированными учебными данными. На этапе тестирования каждый участник получил набор новых тестовых документов, применил свои системы к этим документам и вернул извлеченные шаблоны организаторам MUC. Затем эти результаты сравнивались с набором шаблонов, которые были заполнены вручную, с помощью человека, аналогично тому, как это делается при оценке результатов в других областях NLP.

В целом, семь конференций MUC, проходивших с 1987 по 1998 годы, были посвящены различным задачам извлечения информации. На первых двух конференциях MUC-1 и MUC-2 (1987-1989 годы) основное внимание уделялось автоматизированному анализу военных сообщений, содержащих текстовую информацию о морских наблюдениях и боях, в которых шаблон, подлежащий извлечению, имел десять слотов.

Начиная с MUC-3 задача сместилась на извлечение информации из новостных статей, особый упор был сделан на поиск информации о террористической деятельности, международных фондах, космических аппаратах и ракетах. Структура шаблонов усложнилась. Кроме того, организаторы предоставили учебные и тестовые наборы данных, а также определили меры оценки, которые включали в себя точность и полноту. В 1992 году MUC-4 использовали еще большее количество слотов и пропагандировали использование F-меры для оценки. Начиная с MUC-5, была введена многоязычность системы IE (до этого момента текстовые наборы были на английском языке). Финансовая сфера и новые подзадачи были в центре внимания MUC-6 в 1995 году. Задачи охватывали распознавание именованных сущностей, заполнение шаблонов, разрешение кореференции, неоднозначность смысла слова и синтаксическое структурирование аргумента предиката. В 1998 году в MUC-7 были включены документы об авиакатастрофах на разных языках. Новая задача касалась взаимоотношений между субъектами.

Темы, которые были использованы на конференциях MUC, показывают непрерывный переход от военных интересов к гражданским. В таблице 2.1 представлены основные темы конференций.

Таблица 2.1: Конференции MUC.

MUC	Год	Тип сообщения	Тематика
MUC-1	1987	Военное	Флот
MUC-2	1989	Военное	Флот
MUC-3	1991	Новостное	Террористическая деятельность
MUC-4	1992	Новостное	Террористическая деятельность
MUC-5	1993	Новостное	Корпоративные совместные предприятия
MUC-6	1995	Новостное	Ведение переговоров по трудовым спорам
MUC-7	1997	Новостное	Авиакатастрофы и запуски ракет

На смену конференциям MUC пришла программа «Автоматического извлечения контента» (англ. *Automatic content extraction, ACE*) [11] [12], которая является продолжением инициативы MUC. Программа ACE направлена на поддержку развития автоматизированных технологий извлечения контента для автоматической обработки естественного языка. Программа ACE определила новые и более сложные задачи IE, сосредоточенные вокруг извлечения сущностей, отношений и событий. В частности, возросшая сложность обусловлена: включением различных источников информации, качеством вводимых данных, внедрением более детализированных типов объектов.

Обе инициативы MUC и ACE имеют центральное значение для области IE, поскольку они предоставили наборы корпусов, которые стали доступны исследовательскому сообществу для оценки систем и подходов IE.

К историческому развитию в области извлечения информации можно подходить с разных точек зрения. До конца 1980-х годов системы извлечения информации, основанные на правилах и шаблонах, доминировали в области IE. Даже когда были опубликованы первые подходы, основанные на машинном обучении, они не могли конкурировать с разработанными системами. В конце 1990-х годов подходы основанные на машинном обучении начали завоевывать позиции. Были опубликованы подходы, основанные на наивных байесовских классификаторах и скрытых Марковских моделях. После 2000 года в научных исследованиях стали доминировать модели, основанные на статистических классификаторах и вероятностных графических моделях. В настоящее время количество публикаций о подходах на основе правил снова увеличивается.

С 1995 года ежегодно проводится российская конференция «Диалог» [13], которая посвящена компьютерной лингвистике и интеллектуальным технологиям. Конференция посвящена различным задачам создания инструментов для анализа и моделирования систем применительно к русскому языку. К основным направлениям конференции относятся: семантический анализ, построение формальных моделей русского языка, лексикография, методы оценки систем анализа текста, корпусная лингвистика, извлечение знаний из текста, машинный перевод, анализ и синтез речи.

2.2. Проблематика многоязыковых систем

В контексте возникающих в настоящее время новых видов крупномасштабных корпусов, IE приобретает новые размеры и переосмысливает себя. На протяжении более десяти лет основная часть исследований была посвящена извлечению информации на английском языке. Растущее количество текстовых данных, доступных на других языках, привело к смещению акцента на другие «неанглийские» методы IE. Наибольшее внимание в неанглийских системах IE сосредоточено на решении задачи распознавания именованных сущностей. Относительно мало научных работ об извлечении информации более высокого уровня, например об извлечении отношений, фактов или событий.

IE на других языках, кроме английского, в целом сложнее, и производительность неанглийских систем IE, как правило, ниже. Это связано с отсутствием основных компонентов NLP и базовых лингвистических ресурсов для многих языков. Прежде всего сложность обусловлена различными лингвистическими явлениями, которые не существуют в английском языке, например:

- Отсутствие пробелов, которые рассматриваются как разделители токенов. Например в китайском языке.
- Морфологический анализ усложняют сложные слова, которые имеют в своем составе два или более корня. Это характерно, например, для немецкого языка.
- Нормализацию может усложнять сложное склонение собственных имен. Это типично для славянских языков.
- Свободный порядок слов и богатая морфология, что усложняет извлечение отношений.

и так далее.

2.3. Существующие системы и модули IE

Существуют различные программные инструменты для построения систем извлечения информации. Рассмотрим наиболее популярные из них, представляющиеся свободно:

- **GATE** [14] – бесплатный набор инструментов для многих задач обработки естественного языка, в том числе извлечения информации на различных языках.
- Библиотека **Apache OpenNLP** [15] представляет собой набор инструментов на основе машинного обучения для обработки текста на естественном языке.
- **DBpedia Spotlight** [16] – это инструмент с открытым исходным кодом на Java/Scala, который можно использовать для распознавания именованных сущностей и разрешения кореференции.
- **Stanford CoreNLP** [17] [18] – программное средство, которое предназначено для создания приложений анализа текстов на естественном языке.
- **Machine Learning for Language Toolkit (Mallet)** [19] – это Java-пакет для различных задач обработки естественного языка, включая извлечение информации.

Также существуют программные средства реализации задач IE для русского языка:

- **Томита-парсер** [20] – это инструмент для извлечения структурированных данных (фактов и объектов) из текста на естественном языке.
- **PullEnti** [21] – программный пакет, осуществляющий извлечение именованных сущностей из неструктурированных текстов и другие процедуры обработки текста.

2.4. GATE

General Architecture for Text Engineering (GATE) – это набор Java-инструментов, первоначально разработанный в университете Шеффилда в 1995 году и в настоящее время используемый во всем мире сообществом ученых, преподавателей и студентов для многих задач обработки естественного языка, включая извлечение информации на различных языках. GATE является свободным программным обеспечением с открытым исходным кодом.

GATE включает в себя: интегрированную среду разработки GATE Developer в комплекте с широко используемой системой извлечения информации и набором других плагинов; веб-приложение GATE Teamware – среда совместной работы; GATE Embedded – библиотека объектов и т.д. На рисунке 3 представлено окно GATE Developer.

GATE имеет в своем составе систему ANNIE (англ. *A Nearly-New Information Extraction System*), в которой реализованы готовые решения для токенизации (ANNIE English Tokenizer), тегирования (ANNIE POS-Tagger), разделения текста на предложения и высказывания (ANNIE Sentence Splitter), распознавание и извлечение именованных сущностей (ANNIE NE Transducer), разрешение кореференции (ANNIE OrthoMatcher). Разработчики ANNIE используют для работы регулярные выражения на языке JAPE.

Фрагментам текста приписываются аннотации, с помощью различных модулей системы. Грамматика на языке JAPE состоит из набора правил. Каждое правило состоит из двух частей. Левая сторона правила состоит из описания шаблона, который идентифицирует фрагмент текста по его аннотациям. Правая сторона состоит из операторов, которые манипулируют аннотациями найденного фрагмента. Таким образом, левая часть правила - это шаблон, правая - действие.

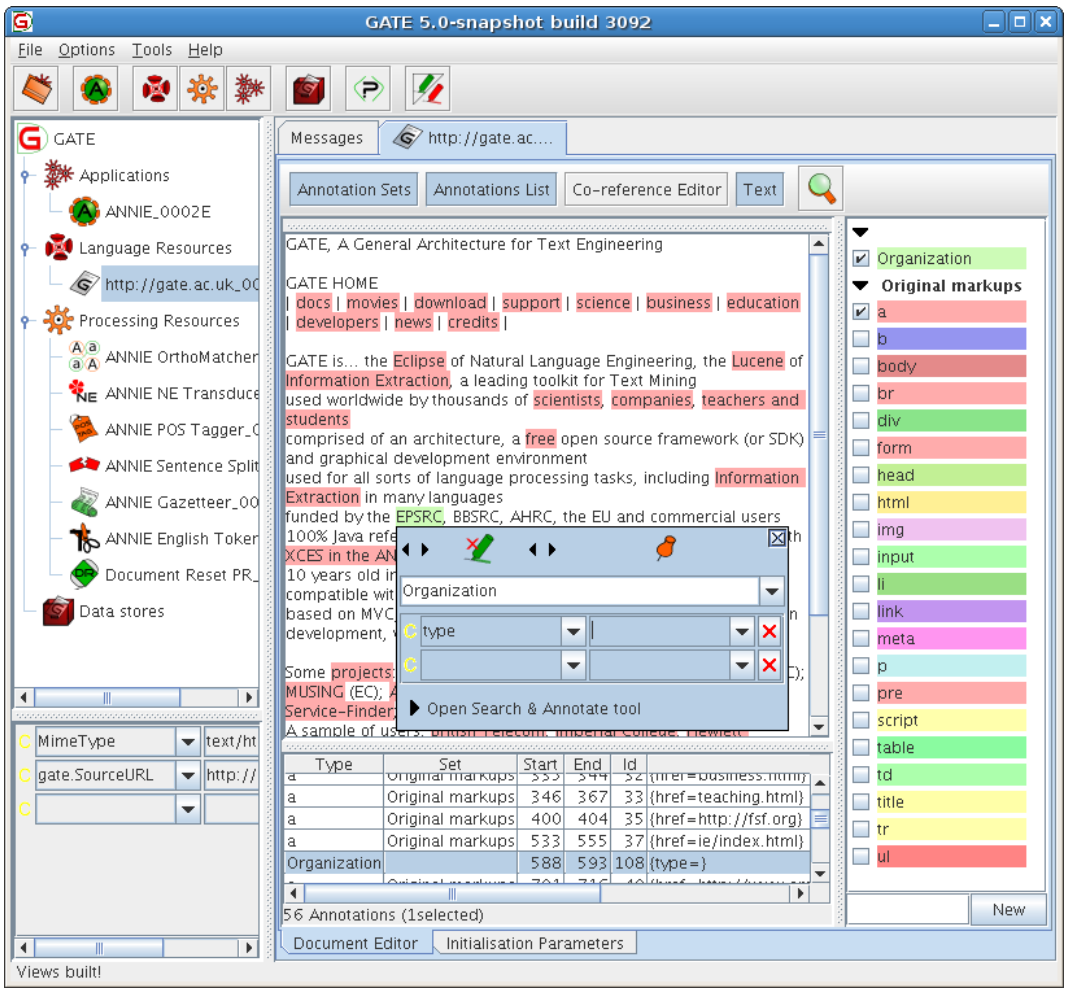

Рис. 3: Рабочее окно интегрированной среды разработки GATE Developer.

2.5. Apache OpenNLP

Библиотека Apache OpenNLP представляет собой набор инструментов на основе машинного обучения для обработки текста на естественном языке. Поддерживает наиболее распространенные задачи NLP, такие как: токенизация, сегментация предложений, POS-тегирование, извлечение именованных сущностей, синтаксический анализ и разрешение кореференции.

Библиотека Apache OpenNLP содержит несколько компонентов, позволяющих построить систему обработки естественного языка. Эти компоненты включают в себя: детектор предложений (SentenceDetector), токенизатор (Tokenizer), поиск сущностей (TokenNameFinder), классификатор документов, POS-тегирование (POSTagger) и так далее. Компоненты позволяют выполнять соответствующие задачи обработки естественного языка, способны обучить и оценить модель. Каждое из этих средств до-

ступно через свой программный интерфейс приложения (англ. *Application Programming Interface, API*). Кроме того, для удобства экспериментов и обучения предусмотрен интерфейс командной строки (англ. *Command Line Interface, CLI*).

2.6. DBpedia

DBpedia [16] - это проект, направленный на извлечение структурированной информации из данных на основе свободной энциклопедии Википедия [7]. DBpedia позволяет пользователям запрашивать отношения и свойства ресурсов Википедии, включая ссылки на другие связанные наборы данных. На рисунке 4 представлена структурированная инфорамция о Владимире Маяковском в DBpedia.

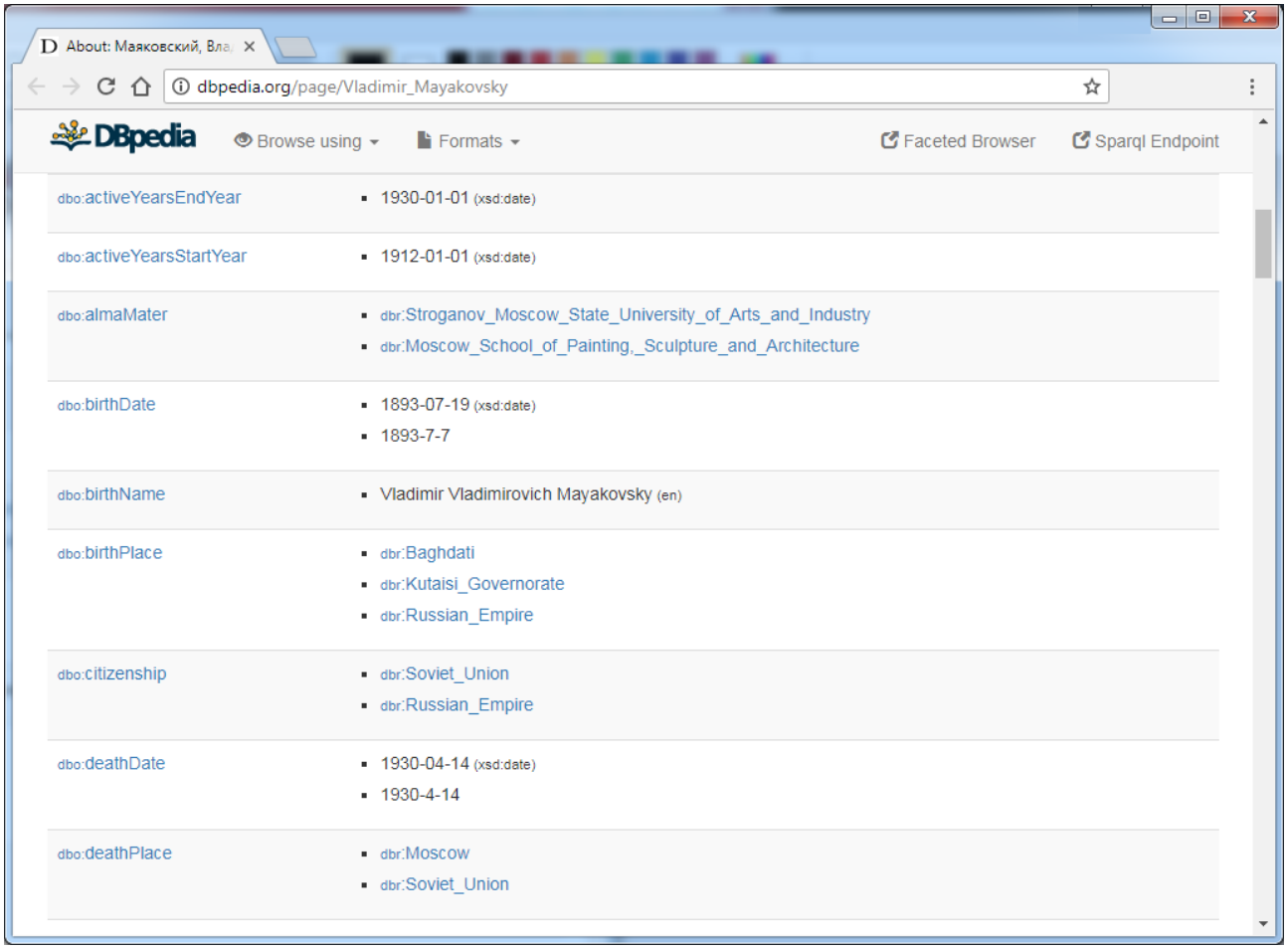

Рис. 4: Страница о Владимире Маяковском в DBpedia.

База данных DBpedia содержит в себе более 4,5 миллионов различных понятий [22], включая такие как: имена людей, географические названия, организации, видеоигры и так далее.

DBpedia использует модель RDF (англ. *Resource Description Framework*

– среда описания ресурса). RDF используется в качестве общего метода для концептуального описания или моделирования информации, реализованной в веб-ресурсах. Модель данных RDF основана на идеи представления высказывания о ресурсе в виде выражения «Субъект – Предикат – Объект», которое называется триплетом. Субъект обозначает ресурс, а предикат обозначает черты или аспекты ресурса и выражает связь между субъектом и объектом. Например, один из способов представления понятия «Вася сын Екатерины Петровны» в RDF - это триплет: субъект – «Вася», предикат – «являться сыном», и объект – «Екатерина Петровна». Множество различные RDF-триплетов представляют собой ориентированный граф. В DBpedia содержится более 3 миллиардов триплетов [22].

DBpedia Spotlight - это инструмент для аннотирования упоминаний ресурсов DBpedia в тексте. DBpedia Spotlight решает задачу выявления и распознования именнованых сущностей, а также разрешения кореференции. DBpedia Spotlight является общедоступной в качестве веб-сервиса и API на языках Java / Scala. В дистрибутив DBpedia Spotlight входит плагин jQuery, который позволяет разработчикам аннотировать страницы в Интернете.

2.7. Stanford CoreNLP

Stanford CoreNLP представляет собой набор инструментов для анализа естественного языка человека. С помощью данных средств возможно нахождение основы слова, части речи, дат, временных промежутков и числовых величин. Также инструменты позволяют разметить структуру предложения в терминах фраз и синтаксических зависимостей, указать, какие фразы относятся к тем или иным сущностям. Возможно осуществление анализа тональности текста, извлечение имен и отношений. На рисунке 5 представлены примеры решения задач распознавания именованных сущностей и разрешения кореференции.

Stanford CoreNLP включает в себя различные модули:

- Stanford Named Entity Recognizer – модуль, для распознавания именованных сущностей.
- Stanford Relation Extractor и Stanford OpenIE – модули для извлечения отношений и фактов из текста.
- Stanford Pattern-based Information Extraction and Diagnostics – модуль для итеративного построения шаблонов для решения задачи IE.

Рис. 5: Решение задач NER и CO.

Используемые модули написаны на языке Java. Каждое из средств доступно для разработчиков через API и CLI.

2.8. Томита-парсер

Томита-парсер – инструмент, созданный компанией «Яндекс». Используется для извлечения и структурирования информации из текстов на естественном языке. Извлечение данных осуществляется с помощью шаблонов (контекстно - свободных грамматик, которые называются КС-грамматиками) и с использованием словарей ключевых слов. Парсер включает в себя следующие модули: токенизатор, сегментатор и морфологический анализатор. У Томита-парсера отсутствует визуальная среда разработки, для разработчиков доступно через CLI.

Три основных компонента парсера: набор грамматик, газеттир и описание фактов.

Газеттир – словарь ключевых слов, который используют грамматики. Каждая из статей газеттира задает различные варианты слов, которые объединены общим свойством. Примером может служить газеттир «Города», в котором содержатся различные цепочки, например «Москва», «Санкт-Петербург», а также различные варианты, такие как «Москвы», «Москву» и так далее.

Грамматика – это множество правил, которые написаны на языке КС-грамматик. Грамматика всегда запускается в пределах одного предложения. Правила для грамматик состоят из двух частей и разделяются знаком «–>». Левая часть правила содержит нетерминал, в правой части стоят терминалы и нетерминалы, а также дополнительные условия. Если

находится цепочка, удовлетворяющая правилу, то ее значение приписывается левой части правила. К терминалам в Томита-парсере относятся: Noun (существительное, слово с граммемой «S»), Verb (глагол, слово с граммемой «V»), Word (любое слово на русском языке), AnyWord (любая последовательность символов без пробелов) и так далее. Нетерменалом является все, что можно построить из терминалов.

Отметим, что одному слову могут соответствовать различные терминалы. Например, если подать на вход цепочку «стекло», то парсер вернет значение [Verb, Noun], так как слово «стекло» может рассматриваться в качестве существительного, которое обозначает прозрачное вещество, так и в качестве глагола, например «что-то стекло со стола».

Факты – построеные в ходе работы парсера таблицы с полями. В грамматике указывается, какие цепочки текста должны быть записаны в поля факта. Описание типов фактов осуществляется в отдельном файле.

Для создания проекта необходимы следующие исходные файлы:

- config.proto – конфигурационный файл, в котором содержится информации о других файлах проекта, о том где они располагаются и как их интерпретировать. Формат: Protobuf.
- dic.gzt – корневой словарь, который содержит в себе перечисление всех используемых в работе парсера грамматик и словарей. Формат: Protobuf и Gazetteer.
- grammar.cxx – грамматика, файл необходим в случае, если в проекте используются грамматики. Таких файлов может быть несколько. Формат: язык описания грамматик.
- fact_types.proto – описание типов фактов, файл необходим в случае, если в проекте извлекаются факты. Формат: Protobuf.
- kw_types.proto – описания типов ключевых слов, файл необходим в случае, если в проекте создаются новые типы ключевых слов. Формат: Protobuf.

Рассмотрим общий вид правил:

$$S -> \ S_1 < Q_1 > \ S_2 < Q_2 > \ \ldots \ S_n < Q_n >;$$

В левой части правила стоит нетерменал S, в правой части терминалы S_1, S_2, ..., S_n и налагающиеся на них дополнительные условия Q_1, Q_2, ..., Q_n. Правило заканчивается знаком «;».

Рассмотрим простое правило для грамматики:

$$P -> \ Adj \ Noun; \tag{2.1}$$

В левой части стоит нетерминал P, в правой части стоят терминалы Adj (прилагательное, причастие, порядковое числительное, или местоименное прилагательное: слово с граммемой «A», «partcp», «ANUM», «APRO») и Noun (существительное). При помощи этого правила извлекаем подцепочки, которые состоят из прилагательного и идущего за ним существительного. Таким образом, при срабатывании этого правила для цепочки «Сегодня было холодное утро» в левую часть правила P записывается «холодное утро».

При написании грамматик на извлекаемые цепочки текста можно наложить дополнительные ограничения, которые называются пометами. Пометы записываются справа от терминала и нетерминала в скобках <>. В качестве ограничений могут быть: определённые словари, грамматические ограничения, согласования слов, регулярные выражения, регистр букв, кавычки и так далее.

Дополним написанное выше правило (2.1) с помощью помет:

$$P -> \ Adj < gnc-agr[1] > \ Noun < gnc-agr[1] >;$$

Помета <gnc-agr> требует согласования слов по роду, числу и падежу. Извлекаются цепочки согласованных между собой прилагательного и существительного. Число в скобках [ ] является идентификатором и указывает, какие два слова должны быть согласованы. Таким образом, для цепочки «Сегодня было холодный утро» правило не сработает, в отличие от приведенного выше примера «Сегодня было холодное утро».

2.9. Выводы

В настоящей главе было рассмотрено историческое развитие задачи извлечения информации и произведен обзор существующих систем и модулей для решения задач IE.

Задача IE берет своё начало с 80-х годов двадцатого века. Важную роль в развитии области IE сыграли конференция MUC и пришедшая ей на смену конференция ACE. За время проведения конференции MUC бы-

ли сформулированы критерии оценки эффективности работы систем IE, а также созданы наборы корпусов документов, которые стали доступны исследовательскому сообществу.

Более десяти лет основная часть исследований была посвящена извлечению информации из текстов на английском языке. В целом, IE для «неанглийских» языков сложнее, из-за различных лингвистических явлений, которые не существуют в английском языке: отсутствие пробелов, сложность образования слов и так далее. С 1995 года проводится ежегодная конференция «Диалог», которая посвящена компьютерной лингвистике для русского языка.

Существует различные программные инструменты для построения систем извлечений информации, такие как: GATE, Apache OpenNLP, DBpedia Spotlight, Stanford CoreNLP, Томита-парсер. Большинство из них способны решать задачи: токенизации, сегментации предложений, POS-теггирования. Некоторые средства также способны решать задачи распознавания именованных сущностей и разрешения кореференции. Большинство систем не поддерживают русский язык, или поддерживают только частично.

Для работы с русским языком подходит Томита-парсер. Парсер включает в себя несколько компонентов: токенизатор, сегментатор и морфологический анализатор. Достоинствами Томита-парсера являются: поддержка русского языка, свободное распространение, обширная и доступная документация, мощный морфологический анализатор.

Глава 3. Обработка естественного языка

Естественным языком называется любой язык, который использует человек. Язык может принимать различные формы, такие как речь, жесты или рукописный текст. Целью обработки естественного языка является построение такой системы, которая бы «понимала» человеческий язык и могла бы управлять полученными данными.

3.1. Развитие задачи NLP

Исторически сложилось так, что обработка речи и языка рассматривалась очень по-разному в информатике, электротехнике, лингвистике. Из-за этого разнообразия обработка речи и языка охватывает ряд несовпадающих, но перекрывающихся областей различных дисциплин: компьютерная лингвистика в лингвистике, обработка естественного языка в информатике, распознавание речи в электротехнике, компьютерная психолингвистика в психологии.

Считается, что история обработки естественного языка берет начало во второй половине двадцатого века. В 1950 году Алан Тьюринг опубликовал статью под названием «Вычислительная техника и интеллект», в которой в качестве критерия интеллекта был предложен так называемый тест Тьюринга [23].

В 1960-х годах была разработана достаточно успешная система обработки естественного языка: ELIZA [24]. Это система написана Джозефом Вайзенбаумом между 1964 и 1966 годами и является симуляцией психотерапевта. ELIZA моделировала разговор, используя сопоставление шаблонов и методологию замены, которая давала пользователю иллюзию понимания со стороны программы. Например при получении текста «У меня болит голова» ELIZA переформулировала его в «Почему вы говорите, что у вас болит голова?». На рисунке 6 представлено диалоговое окно ELIZA.

В 1970-е годы многие программисты начали писать «концептуальные

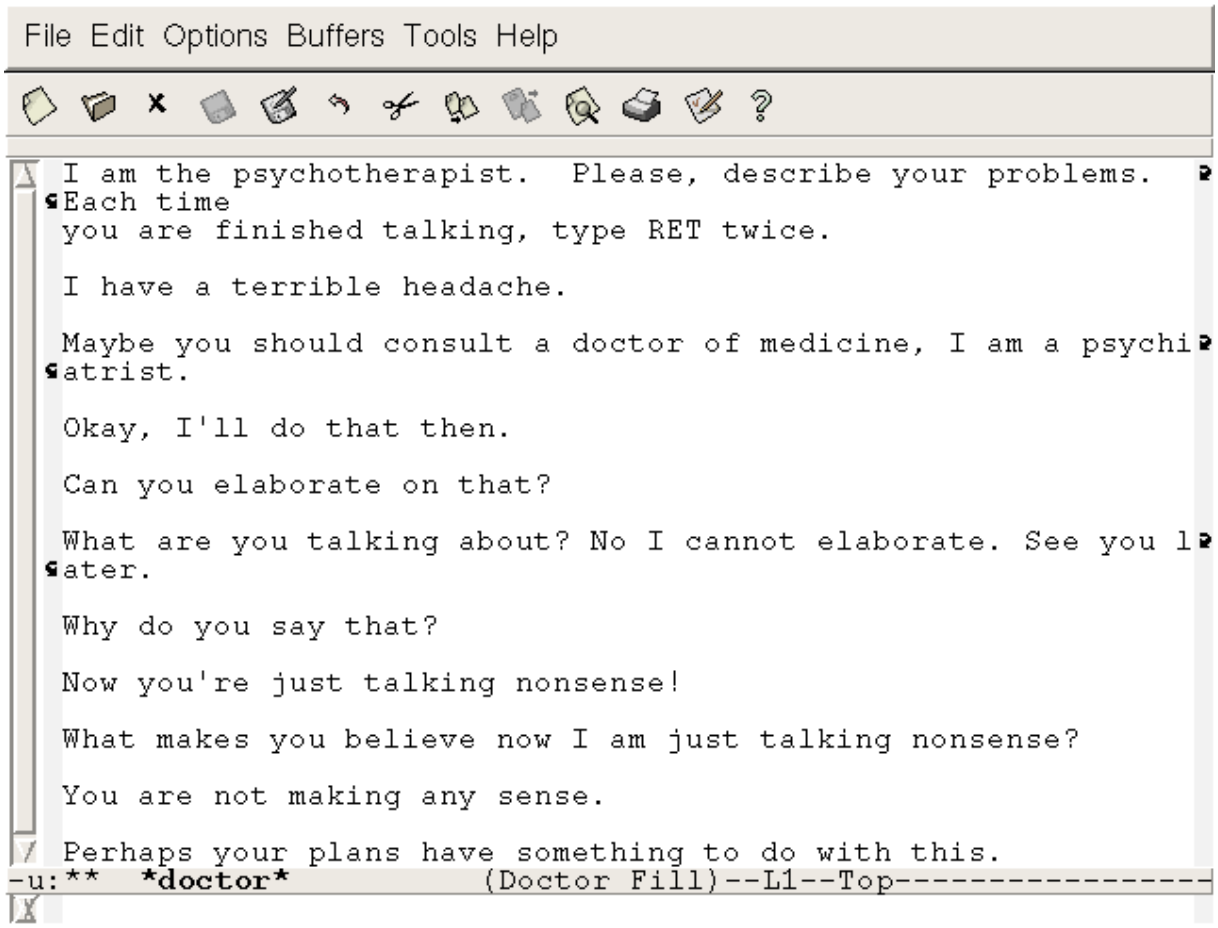

Рис. 6: Диалоговое окно ELIZA.

онтологии», которые структурировали реальную информацию в понятные для компьютера данные. До 1980-х годов большинство систем обработки естественного языка основывались на сложных наборах рукописных правил. Однако, начиная с конца 1980-х годов, произошла революция в обработке естественного языка с внедрением алгоритмов машинного обучения для обработки языка [25]. Начиная с 1990-х NLP стремительно развивается. Огромное количество текстов, наводнивших Интернет, стимулировали работу над задачами по управлению этими данными, в частности путем извлечения информации и автоматического аннотирования [26].

В двадцать первом веке, благодаря увеличению вычислительной мощности, низкой стоимости и доступности компьютерной памяти, активно развиваются разработки в области NLP. Современные разработки породили множество новых и сложных исследовательских тем. Начиная от машинного чтения текста, в котором компьютеры могут «читать» и «понимать» текст, до распознавания речи. Компьютеры по всему миру становятся все более и более продвинутыми, например: Watson - это компьютерная система ответов на вопросы, способная отвечать на вопросы, заданные на естественном языке, разработанная в IBM в 2006 году [27]. Watson выслу-

шивает вопрос, «понимает» его, используя современные методы NLP, проводит поиск по массивной базе данных, и находит наиболее верный ответ.

3.2. Проблематика обработки естественного языка

Естественный язык является нашим основным средством коммуникации и может быть охарактеризован как набор символов, которые мы организуем в структурной форме в высказывания. Данные высказывания передают определенный смысл. Структура высказываний называется его синтаксисом, а смысл – семантикой. Области компьютерной лингвистики и NLP предлагают большое разнообразие представлений и формализмов для описания языка. Фактически, вопрос о том, как концептуализировать язык, тесно связан с исследованиями в области человеческого познания, математической логики и философии.

Две проблемы затрудняют обработку естественных языков:

- Уровень неопределенности, который существует в естественных языках.
- Сложность семантической информации, которая содержится даже в простых предложениях.

Обычно языковые процессоры имеют дело с большим количеством слов, многие из которых имеют альтернативное значение. Сложность, в том числе, заключается в неравномерности языка и в различных видах неоднозначности, которые возникают в тексте. Рассмотрим некоторые примеры:

1. В высказывании «Мама мыла стекло» существует проблема определения части речи слов. Слово «мыла» может быть использовано как в качестве глагола, который обозначает действие «мыть», так и в качестве существительного, которое обозначает предмет «мыло». Аналогичная проблема при обработке слова «стекло», которое может быть распознано как существительное, которое обозначает предмет «стекло», но также может рассматриваться как глагол, обозначающий действие «стекать». Эти сложности вызывают неоднозначность, которая должна быть устранена во время синтаксического анализа текста. Любой механизм разбора текста должен быть способен исследовать различные синтаксические конструкции фраз и иметь возможность отслеживать и перестраивать их по мере необходимости.

2. Рассмотрим два высказывания: «Собака играет с мячом» и «Собака играют с мячом». Проблема со вторым высказыванием заключается

в том, что между субъектом «собака» и предикатом «играют» не существует согласования по числу. При написании правил для обработки языка и извлечения информации разработчик должен писать выразительные шаблоны, в которых необходимо указывать проверки на такие аномалии. Также необходимо описывать действия, которые должны осуществляться в таких случаях. Полезны механизмы предупреждения сбоев в обработке текста. В случае проблем с согласованием между субъектом и предикатом может быть уместным сигнализировать предупреждение. Предупреждение помечает вложенную фразу как потенциально ошибочную, но не отвергает ее полностью.

3. Рассмотрим два высказывания: «Ребёнок пнул мяч» и «Ребёнок пнул стену». В первом высказывании смысл заключается в том, что «ребёнок» выполнил действие «пнул», которое привело в движение предмет «мяч». Во втором высказывании смысловая нагрузка иная - «ребёнок» выполнил действие «пнул», то есть привел в движение «ногу», но слово «нога» не указано явно в предложении. В данном случае имеет место неоднозначность. В решении этой неоднозначности может помочь знание о том, что мячи являются подвижными объектами (и часто перемещаются с помощью ноги в качестве инструмента), а стены – статическими объектами.

Сложности в обработке языка также связаны с особенностями и различными лингвистическими явлениями языков. Рассмотрим проблемы, которые имеют место при анализе русскоязычных текстов:

- **Свободный порядок слов.** При анализе высказываний «Петя любит Машу» и «Маша любит Петю» результат будет различный. Обе цепочки состоят из одних и тех же лексем, но в первом случае, акцент делается на то, что именно «Петя» «любит Машу». Во втором же случае, ударение идет на то, что именно «Маша» осуществляет действие «любить» по отношению к «Петя».
- **Раскрытие анафор.** Рассмотрим два высказывания «Петя принес Маше тарелку пастилы, потому что она его просила» и «Петя принес Маше тарелку пастилы, потому что она вкусная» – данные высказывания имеют схожую структуру. В левой части высказываний упоминаются два лица – «Петя» и «Маша». В первом случае местоимение «она» в правой части указывает на «Машу», но во втором случае местоимение «она» относится к слову «пастила». Процесс раскрытия анафор в русском языке является трудоемким и сложным.
- **Пунктуация.** В зависимости от расположения запятых в предложении может меняться смысл высказывания. Рассмотрим крылатое выражение «Казнить нельзя помиловать». Смысл фраз «казнить нельзя,

помиловать» и «казнить, нельзя помиловать» противоположен.

- **Снятие омонимии.** Омонимы – это одинаковые по написанию, но различные по значению слова. В качестве примера можно привести цепочку «Маша с косой работала в поле косой». В данном случае слово «коса» рассматривается как вариант прически «коса», и как обозначение инструмента «коса».

Представленные примеры показывают, какие возникают сложности при обработке естественного языка – текста. Системы NLP вынуждены решать проблемы, аналогичные тем, что были проиллюстрированы выше.

3.3. Общие этапы обработки текста

Язык может быть определен как набор символов. Символы объединяются и используются для передачи информации или трансляции информации. В процессе обработки естественного языка выделяют четыре основных этапа, которые представлены на рисунке 7. В реальных системах эти этапы редко происходят как отдельные, последовательные процессы. В некоторых системах часть этапов отсутствует, объединяется или вводятся дополнительные шаги обработки текста.

- **Токенизация и сегментация** – разбиение текста на токены (слова) и предложения. Переход от символов к предложениям и к словам.
- **Морфологический анализ** – осуществляет анализ словоформ и поиск их лексем. Переход к основам слова.
- **Синтаксический анализ** – анализ структурных отношений и связей между словами.
- **Семантический и прагматический анализ** – анализ смысловой составляющей текста.

Рис. 7: Этапы обработки текста.

3.4. Токенизация и сегментация

В лингвистическом анализе текстов на естественном языке необходимо четко определять, что представляет собой слово и предложение. Определение этих единиц порождает различные задачи, но данные задачи являются нетривиальными, учитывая разнообразие человеческих языков и

систем письменности. Слова и предложения, выявленные на данном этапе, являются основными единицами, принятыми для дальнейшей обработки текста.

Токенизация – это процесс разбиения последовательности символов в тексте путем определения границ слова, где заканчивается одно слово, и начинается другое. В компьютерной лингвистике идентифицированные таким образом слова называются токенами. Данный этап важен и представляет сложность в письменных языках, где в системе письма нет явных границ слов.

Токенизация слов может показаться простой, например, в русском языке, где слова разделяются специальным символом – пробелом. Но в таких языках как китайский, японский нет такой системы границ слов. К тому же, разделение слов только пробелом недостаточно. Рассмотрим пример: «Маша мыла посуду, она устала.» Если рассматривать слова, как то, что находится между границ – пробелов, то тогда при обработке данного примера возникнут слова «посуду,», «устала.». Эти ошибки можно устранить, рассматривая пунктуационные знаки как дополнение к пробелу. Однако это может привести к новым ошибкам, так как неправильно будут идентифицированы следующие слова: «т.д.», «spbu.ru», «36.6», «21/09/93» и так далее. Алгоритмы токенизации также способны разделять многословные выражения, такие как «Йошкар-Ола», для чего требуется словарь многословных выражений. Это делает токенизацию тесно связанной с задачей обнаружения имен, дат и организаций, то есть с задачей распознавания именованных сущностей.

Сегментация – это процесс разбиения текста на предложения.

Разделение текста на предложения обычно основано на пунктуации. Это связано с тем, что некоторые виды знаков препинания (точки, вопросительные знаки, восклицательные знаки), как правило, обозначают границы предложения. Вопросительные и восклицательные знаки являются относительно однозначными маркерами границ предложений, а символ «.» не всегда означает границу предложения, например «В. Маяковский.». По этой причине разделение текста на предложения и слова, как правило, рассматриваются совместно.

3.5. Морфологический анализ

Морфема – наименьшая значимая языковая единица. Морфемы разделяются на два основных типа: корень и аффикс. Это соответственно основная значимая и вспомогательная части слова. Аффиксы подразделяются на два типа: префиксы и постфиксы, которые располагаются соответственно до и после корня. В русском языке префиксы – это приставки, а постфиксы – это суффиксы и окончания.

Морфологический анализ – это процесс сопоставления словоформам (словам текста) их лексем (словарных форм), а также переход к основным значимым частям слова. Морфологический анализ в первую очередь занимается определением того, как была изменена «базовая» форма слова. Изменение обычно происходит путем добавления префиксов и постфиксов. Например, словоформа «несчастливому» имеет лексему «несчастливый», приставку не-, корень -счаст-, суффикс -лив и окончание -ому.

Характер морфологического анализа в значительной степени зависит от анализируемого языка. В некоторых языках отдельные слова (используемые как глаголы) содержат всю информацию о времени, роде и числе и так далее. В других языках эта информация может быть передана через несколько слов в предложении. Например, в англоязычном предложении «He will have wrote this essay by Monday.» сложная информация о времени передается через вспомогательные глаголы «will» и «have». В русскоязычном предложении «Он напишет сочинение к понедельнику.» информация о времени передается с помощью аффикса глагола «напишет».

Морфологический разбор очень важен. Он играет решающую роль в задаче информационного поиска в морфологически сложных языках, таких как русский или немецкий. В русском языке существует множество словоформ одной и той же лексемы. В качестве примера в таблице 3.1 представлено склонение слова «стол» по падежу и числу. Таким образом, например, задача поиска информации о слове «стол» в тексте должна учитывать всевозможные варианты изменения этого слова.

Таблица 3.1: Склонение слова «стол» по падежу и числу.

падеж	ед. ч.	мн. ч.
Им.	стол	столы
Р.	стола	столов
Д.	столу	столам

В.	стол	столы
Тв.	столом	столами
Пр.	столе	столах

Морфологический анализ – это процесс нахождения составляющих морфем слова. На данном этапе осуществляется POS-тегирование, то есть для каждого слова в тексте определяется часть речи и набор морфологических характеристик.

3.6. Синтаксический анализ

Цель синтаксического анализа состоит из двух частей: проверить, что строка слов (предложение) сформирована корректно, и создать структуру, показывающую синтаксические отношения между разными словами. Синтаксический анализатор делает это, используя словарь определений слов (лексикон) и набор синтаксических правил (грамматика). Простой лексикон содержит только синтаксическую категорию каждого слова, простая грамматика описывает правила, которые указывают только как синтаксические категории могут быть объединены для формирования фраз разных типов. Не все системы NLP требуют полного разбора предложений, то есть проведения полного синтаксического анализа.

Рассмотрим в качестве примера предложение:

«Незнайка совершил путешествие в Солнечный город.»

В таблице 3.2 представлен лексикон, в таблице 3.3 – грамматика.

Таблица 3.2: Лексикон.

Слово	Категория
незнайка	существительное
совершил	глагол
путешествие	существительное
в	предлог
солнечный	прилагательное
город	существительное

Таблица 3.3: Грамматика.

Предложение -> существительное ГруппаГлаг ГруппаСущ
ГруппаГлаг -> глагол существительное
ГруппаСущ -> предлог ГруппаПрилагСущ
ГруппаПрилагСущ -> прилагательное существительное

Представление предложения с помощью лексикона и грамматики приведено на рисунке 8.

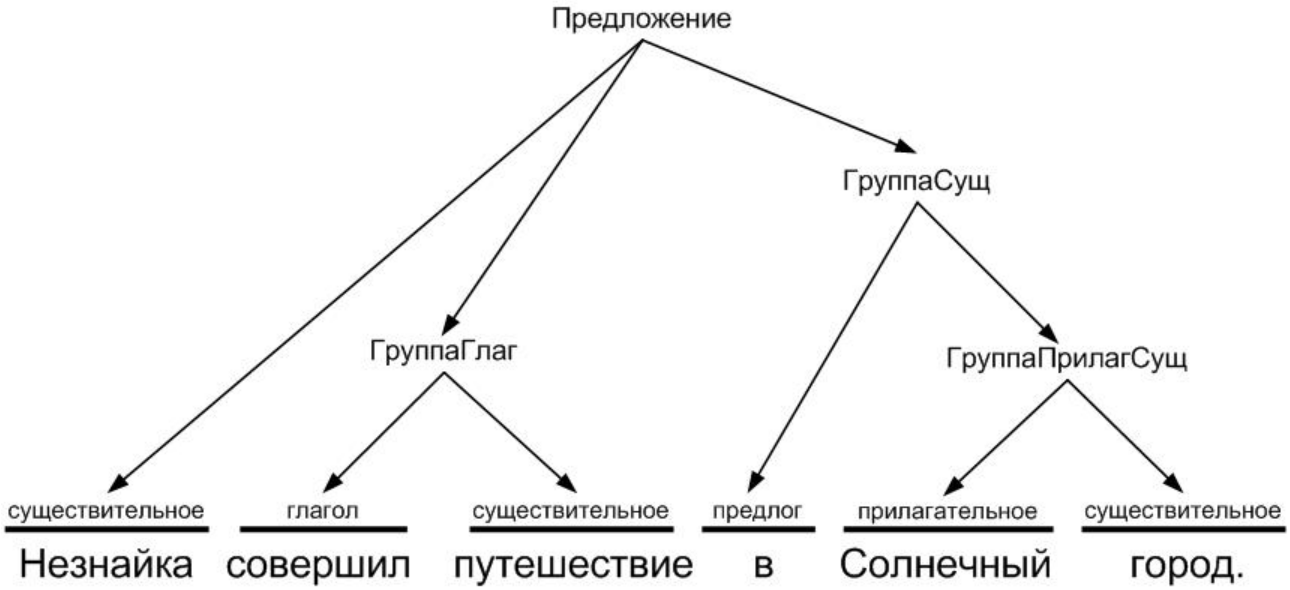

Рис. 8: Синтаксический анализ текста.

3.7. Семантический и прагматический анализ

Конечная цель, как для человека, так и для системы обработки естественного языка, состоит в том, чтобы понять высказывание. «Понимание» высказывания – это сложный процесс, который зависит от результатов предыдущих этапов системы NLP, а также от лексической информации, контекста и здравого смысла; и приводит к тому, что называется семантической интерпретацией в контексте высказывания.

Проектирование семантического интерпретатора предполагает решение тех же проблем, с которыми приходится сталкиваться при построении синтаксического анализатора, в частности, проблемы (семантической) двусмысленности: как распознать предполагаемую семантическую интерпретацию высказывания предложения в конкретном контексте, среди множества возможных интерпретаций этого предложения.

Задача разработки семантического интерпретатора сложна, поскольку между исследователями мало согласия относительно того, какой именно должна быть конечная интерпретация высказывания. Практические системы NLP, как правило, используют семантические представления, предназначенные для конкретной предметной области.

Семантический анализ связывает смысл с изолированными высказываниями (предложениями), прагматический анализ интерпретирует результаты семантического анализа с точки зрения конкретного контекста. Это означает, что в результате семантического анализа высказывания «Незнайка совершил путешествие в Солнечный город» можно получить выражение «Незнайка», которое является именем, а с помощью прагматического анализа можно сделать вывод о том, что «Незнайка» – это литературный персонаж произведений Н.Носова. В некоторых случаях, как и в только что описанном примере, прагматический анализ находит соответствие реальным объектам или событиям, которые существуют в данном контексте, со ссылками на объекты, полученными в ходе семантического анализа.

3.8. Выводы

В настоящей главе описаны основы задачи обработки естественного языка, приведена историческая справка, а также основные этапы работы системы NLP.

Считается, что задача обработки естественного языка берет свое начало с 50-х годов двадцатого века. Проблематика задачи NLP в основном связана с уровнем неопределенности, который существует в языках, и сложностью семантического анализа предложений. В процессе работы с русскоязычными текстами, часто возникают трудности при решении следующих задач: раскрытие анафор, снятие омонимии. Свободный порядок слов и пунктуация в русском языке также усложняют задачу обработки.

В процессе обработки естественного языка условно выделяют четыре основных этапа: токенизация и сегментация - разбиение текста на слова и предложения; морфологический анализ – процесс нахождения морфем слов; синтаксический анализ – анализ структурных отношений и связей между словами; семантический и прагматический анализ – анализ смысловой составляющей текста. В реальных системах эти этапы могут объединяться, исключаться или дополняться.

Глава 4. Методы машинного обучения

Исследователи в области машинного обучения (ML) рассматривают вопрос о том, как сделать машины способными «учиться», то есть чтобы машины имели способность изменять свое поведение таким образом, чтобы в будущем выполнять задачу лучше. В частности, машинное обучение – это метод создания компьютерных программ путем анализа данных. Некоторые системы ML основаны на взаимодействии людей и машин (обучение с учителем), в то время как другие пытаются устранить потребность в человеке при анализе данных (обучение без учителя). ML является естественным результатом пересечения информатики и статистики.

В девяностых годах двадцатого века произошел поворот в компьютерной лингвистике от ручного построения грамматик и баз знаний к частичной или полной автоматизации этого процесса с помощью статистических методов обучения, которые обучаются на больших корпусах естественного языка. Популярность алгоритмов ML обусловлена производительностью аппаратных и программных архитектур, которые позволяют обрабатывать огромные объемы информации.

Методы машинного обучения широко применяются в интеллектуальном анализе данных, в частности, для обнаружения скрытых закономерностей в растущих объемах данных в Интернете. В NLP и TM применение методов ML к обработке массивных объёмов данных на свободном языке способствовало развитию различных методов и приемов для решения проблем, которые возникают при обработке естественного языка.

Отправной точкой для решения большинства задач NLP и TM является поиск и создание структур в неструктурированных данных, что является основной целью методов классификации и кластеризации.

4.1. Задача классификации

Задача классификации предполагает определение, к какому классу (или классам) из заранее заданного набора классов (категорий) относит-

ся анализируемый объект (текстовый документ). Этот процесс аналогичен тому, как книгам в библиотеке присваиваются различные категории.

Одним из подходов к реализации данного процесса является классификация с использованием правил, которые чаще всего формируются человеком. Такой набор правил фиксирует определенную комбинацию ключевых слов, которая указывает на класс. Поскольку обслуживание созданных вручную правил является трудоемким процессом, существует еще один подход к классификации текста, а именно классификация текста на основе методов машинного обучения. В ML критерии принятия решения о классификации документа определяются автоматически на основе данных, полученных на этапе обучения. Обучающие данные должны служить хорошим примером документов каждого класса. Данные документы должны быть размечены. Маркировка данных является более простой задачей, чем написание правил. Этот тип ML называется обучением с учителем, потому что для управления процессом обучения необходим эксперт.

Рассмотрим задачу классификации документа [28]. Пусть $d \in D$, где d – некоторый документ (от англ. *document*), D – множество документов. Пусть $C = \{c_1, c_2, ..., c_n\}$, где c_i, $i = 1, ..., n$ – определенный класс (от англ. *class*). Множество C – задается экспертом вручную. Также существует обучающее множество T (от англ. *training set*) в котором содержатся размеченные документы, т.е. $< d, c > \in D \times C$.

В обучающем множестве, например, может содержаться следующий элемент:

$$< d, c > = <\text{А судьи кто?, Горе от ума}>$$

где d содержит в себе цитату «А судьи кто?» из произведения «Горе от ума» и относится к классу $c =$«Горе от ума».

Задача классификации состоит в поиске функции классификации γ, с помощью обучающего множества T и алгоритма обучения, следующего вида:

$$\gamma : D \to C$$

4.2. Задача кластеризации

В отличии от классификации, кластеризация является наиболее распространенной формой неконтролируемого обучения. Нет человека - эксперта, который должен определить, какие объекты каким классам принадлежат. Алгоритмы кластеризации группируют набор объектов в подмноже-

ство кластеров. Цель алгоритмов состоит в том, чтобы создать кластеры. Внутренние элементы каждого кластера должны быть схожи между собой и при этом должны отличаться от элементов других кластеров [28].

Рассмотрим кластеризацию объектов - точек, с учетом их положения относительно друг друга, которые изображены на рисунке 9.

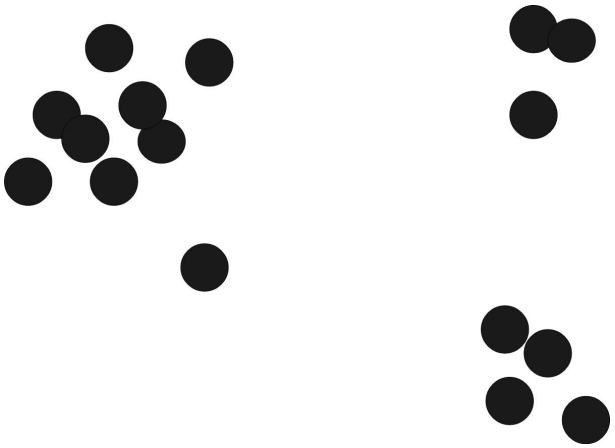

Рис. 9: Исходные данные.

В результате применения алгоритма класстеризации мы можем получить три кластера A, B, C, изображенные на рисунке 10, внутри которых находятся схожие между собой элементы.

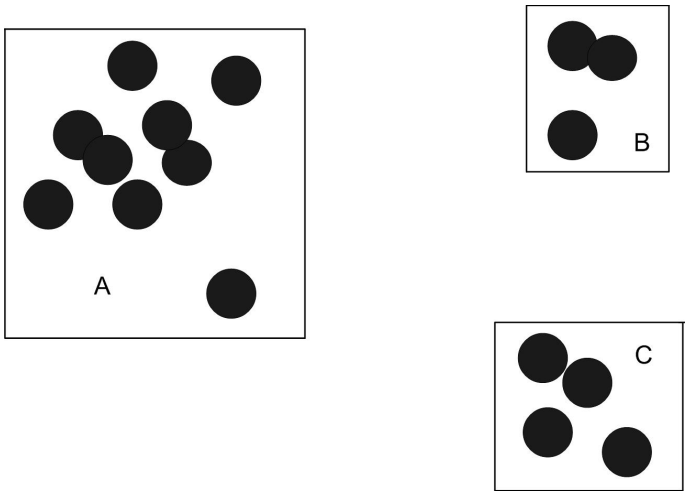

Рис. 10: Кластеры A, B, C.

Пусть $D = \{d_1, d_2, ..., d_m\}$ – множество документов. Пусть $K = \{1, 2, ..., N\}$ – множество идентификаторов кластеров, а N – число, которое обозначает общее количество кластеров (в общем случае заранее неизвестно). Пусть имеется функция $\rho(d_1, d_2)$, которая обозначает степень схожести двух документов $d_1, d_2 \in D$. Задача кластеризации состоит в том, чтобы найти минимально возможное N, при котором каждому элементу $d \in D$ будет поставлен в соответствие $k \in K$, с учетом близости метрики ρ элементов

кластера.

4.3. ML в задаче извлечения информации

Извлечение информации определяется как идентификация определенных сущностей и отношений. С точки зрения машинного обучения это определение соответствует классификации фрагментов текста по заданному классу или метке. Таким образом, существует возможность для представления задачи извлечения информации как задачи машинного обучения. Методы машинного обучения для извлечения информации обычно используют модели с обучением. Параметры моделей оцениваются в соответствии с приведенными аннотированными учебными данными для прогнозирования набора меток или сущностей [2].

Примером может служить задача распознавания именованных сущностей. Рассмотрим входной текстовый файл, в котором содержится размеченный текст. Слова, которые обозначают имена сущностей, помечены «1». Все остальные слова помечены «0». Таким образом, задача извлечения имен из текста может рассматриваться, как процесс классификации слов – отнесение каждого слова к одному из двух классов «1» и «0». В связи с этим, алгоритм классификации можно рассматривать как алгоритм, который используется для того, чтобы определить, какие подстроки текста нужно извлечь, а какие нет.

Простым подходом к классификации является использование наивного байесовского классификатора. В машинном обучении наивные байесовские классификаторы представляют собой семейство простых вероятностных классификаторов, основанных на применении теоремы Байеса со строгими предположениями о независимости между признаками [29]. Несмотря на свою простоту, они широко используются в ML и NLP с удивительным успехом. Наивные байесовские классификаторы предполагают, что влияние значения переменной на данный класс не зависит от значений других переменных. Это предположение называется условной независимостью класса. Предположение сделано для упрощения вычислений и в этом смысле считается наивным. Исследования показали высокую точность и скорость работы этих методов при применении к большим наборам данных.

Маркировка последовательности используется во многих задачах обработки естественного языка. Слова текста рассматриваются как наблюдения, которые имеют маркировку – набор характеристик. В этом случае при-

меняются модели последовательностей: скрытая Марковская модель (англ. *Hidden Markov Model*), Марковская модель с максимальной энтропией (англ. *Maximum Entropy Markov Models*), метод условных случайных полей (англ. *Conditional Random Field*) [4].

4.4. Выводы

В настоящей главе рассмотрено применение машинного обучения к задаче извлечения информации. К задачам, которые решаются с помощью методов машинного обучения, относятся классификация и кластеризация.

Задача классификации документов состоит в том, чтобы соотнести документ из набора к группам или группе документов, которые являются схожими между собой. Суть групп документов может быть различна, они могут являть собой конкретные темы, определенные события или категории. В случае машинного обучения, для решения этой задачи необходимо выполнить обучение с учителем. Для обучения с учителем системе необходимо предоставить обучающую выборку, состоящую из некоторого набора текстовых документов, а также набора классов, к которым эти текстовые документы принадлежат.

В случае кластеризации документов, построенная система должна самостоятельно установить множество различных кластеров, к которым могут быть определены текстовые документы. Данная задача в терминах машинного обучения именуется обучением без учителя.

Задача извлечения информации заключается в идентификации различных предопределенных объектов из текста, например сущностей или отношений. С точки зрения области ML – это соответствует задаче классификации. Таким образом, можно рассматривать IE как задачу ML.

Глава 5. Реализация автоматизированной системы

В данной главе представлена реализации автоматизированной системы определения события международного значения на основе анализа новостных лент на различных языках.

5.1. Условия

Сформулируем условия, которые налагаются на систему определения события:

В качестве исходных данных выступают новостные сообщения различных средств массовой информации. В системе используются источники информации двух типов: новостная лента пользователя и альтернативная лента. Новостная лента пользователя строится на основе популярных русскоязычных СМИ, которые распространяются свободно в Интернете. В качестве альтернативной ленты рассматриваются новостные ленты на английском, французском и немецком языках.

Зачастую новостные статьи состоят из следующих блоков: заголовок, новостное сообщение, дата, дополнительные пометки (категория, тема и т.д.). Функция новостного заголовка заключается в том, чтобы привлечь внимание читателя. Зачастую, в заголовке кратко и ёмко представлена ключевая информация новостной статьи. В рамках данной работы определение события происходит в границах одного предложения, на основе анализа заголовка новостного сообщения на русском языке.

Под «событием» будем понимать набор ключевых слов, которые отвечают на вопрос «кто и что сделал?». Представим событие в виде тройки ключевых слов: Субъект (англ. *Subject*) – Предикат (англ. *Predicate*) – Объект (англ. *Object*). Под предикатом будем понимать действие, которое направлено субъектом по отношению к объекту. Отметим, что в русском языке субъекту в предложении соответствует подлежащее, предикату – сказуемое, объекту – дополнение. Будем считать, что событие идентифи-

цировано, если в тройке <Subject, Predicate, Object> найден хотя бы предикат. Событие считается международным, если найдены статьи о данном событии в альтернативной новостной ленте. При поиске новостей используются ключевые слова, которые переведены на языки альтернативной новостной ленты.

5.2. Архитектура системы

Схематично архитектура системы представлена на рисунке 11.

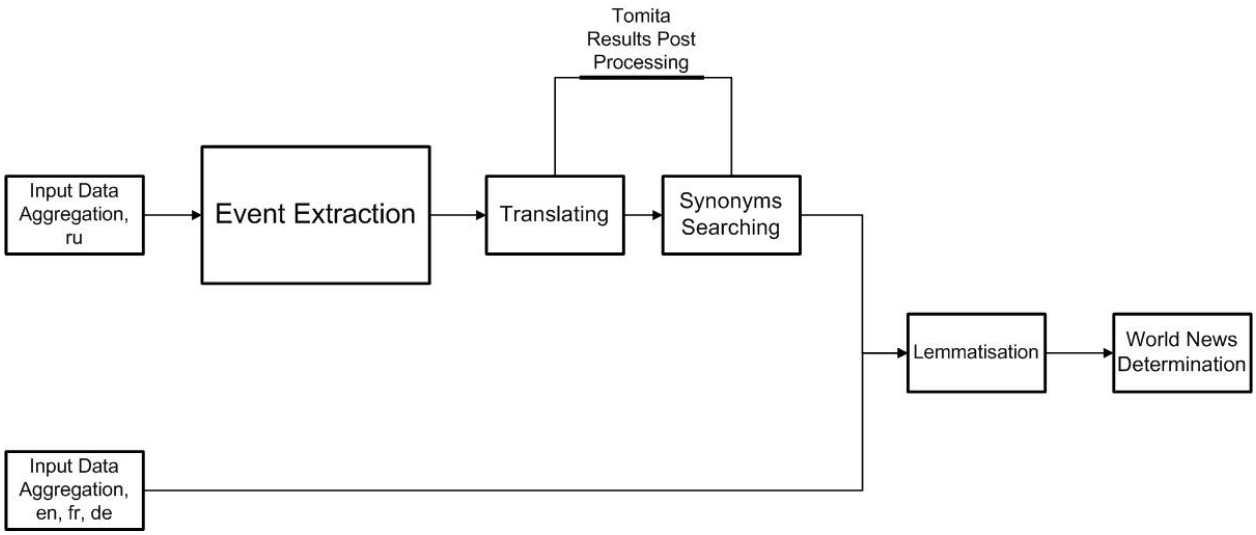

Рис. 11: Архитектура системы.

Система состоит из пяти модулей:

- **Input Data Aggregation.** В данном модуле происходит сбор исходных данных – новостных сообщений.
- **Event Extraction.** Ключевой модуль системы. В данном модуле происходит обработка текста: токенизация, сегментация и морфологический анализ. Извлечение событий происходит с помощью написанных правил на языке контекстно-свободных грамматик, используя Томита-парсер. В результате работы модуля выделяются тройки <Subject, Predicate, Object> в виде ключевых слов для идентификации события.
- **Tomita Results Post Processing.** Модуль постобработки результатов, полученных на предыдущем этапе. Модуль состоит из двух компонент: перевод и поиск синонимов. Производится перевод извлеченных троек с русского языка, на языки альтернативной ленты. Для разрешения кореференции, а также неточности перевода, производится поиск синонимов ключевых слов.

- **Lemmatisation.** Модуль лемматизации. то есть приведение слов к их лемме. Применяется ко всем наборам переведенных ключевых слов, а также к сообщениям альтернативной новостной ленты.
- **World News Determination.** Модуль определения международной новости и предоставления альтернативных новостей. Определение значимости события производится на основе поиска ключевых слов в альтернативной новостной ленте.

5.3. Модуль Input Data Aggregation

В качестве новостной ленты пользователя рассматриваются следующие источники информации – Lenta.ru, ТАСС, Вести.Ru, Издательский дом Коммерсантъ. В качестве аль тернативной: англоязычные – The Guardian, BBC News, The Washington Post, The New York Times, USA Today, The Independent; французскоязычные – Le Figaro, Le Monde, Libération; немецкоязычные – Deutsche Welle, Frankfurter Allgemeine Zeitung, Die Welt.

Сбор информации осуществляется на основе RSS-каналов (англ. *Rich Site Summary* — обогащённая сводка сайта) новостных порталов. RSS – семейство XML-форматов, которые специализируются на описании новостных лент. Эти каналы могут, например, позволить пользователю отслеживать множество различных веб-сайтов в одном агрегаторе новостей. Стандартный формат файла XML обеспечивает совместимость со многими программами. Полный список, использующихся в работе RSS каналов, представлен в Приложении 2. Пример xml-файла, полученного из RSS канала, представлен на рисунке 12.

За счет того, что RSS – это общепринятый и стандартизированный формат передачи информации, все получаемые на выходе XML файлы имеют идентичную структуру, вне зависимости от источника распространения. Для всех XML файлов можно применять одинаковый механизм десериализации. Десериализация – процесс восстановления исходной структуры данных с помощью последовательности битов. В данной работе использовался пакет System.ServiceModel.Syndication, входящий в состав .NET Framework 4.7.2, предназначенный для обработки XML, используемых в RSS рассылках. Исходный код программы представлен в Приложении 3.

В результате десериализации мы получаем список объектов, каждый из которых представляет конкретную новость из произвольной RSS ленты. Так как нам нужна не вся информация, содержащаяся в таком объекте, то для дальнейшей работы отделяем только следующий ряд необходимых компонент:

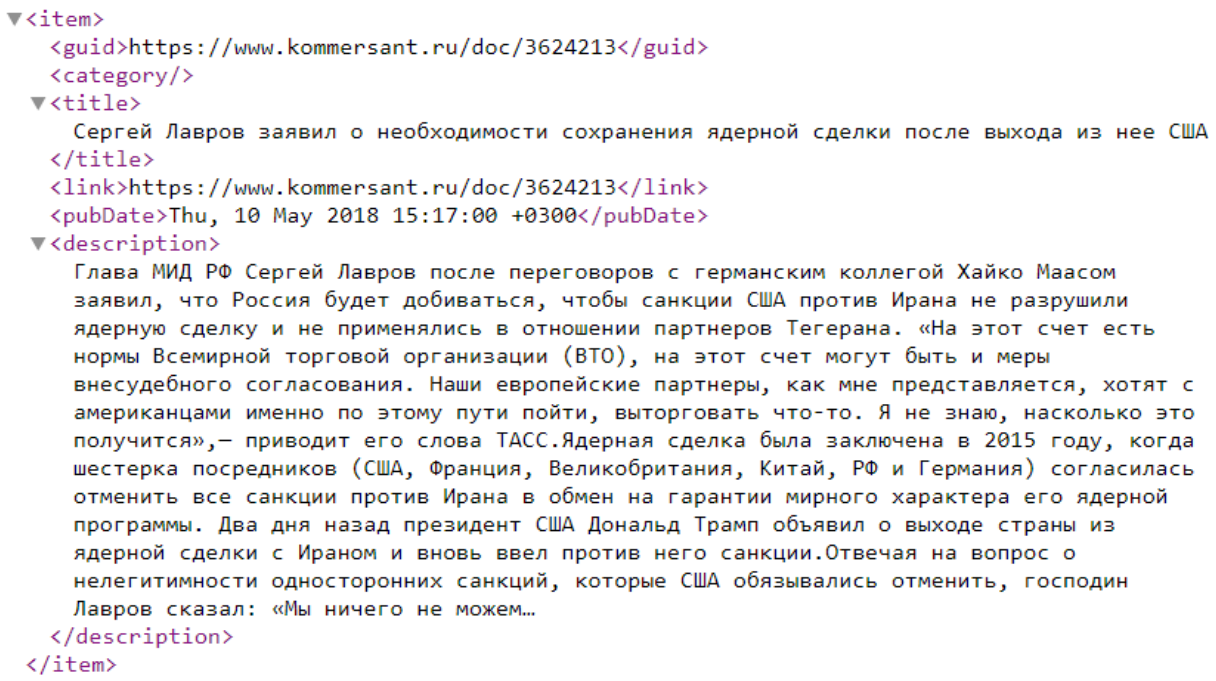

Рис. 12: XML-файл, полученный из RSS-канала новостного источника.

1. Уникальный идентификатор новости.
2. Заголовок новости.
3. Содержание новости.
4. Список тегов - пометок.
5. Ссылка на статью на новостном портале в сети Интернет.
6. Дата и время публикации.

Для последующего извлечения событий нам нужны только заголовки полученных новостей. Все русскоязычные заголовки собираются в один текстовый файл. Оставшаяся информация потребуется нам на следующих этапах работы системы.

В качестве обучающего множества было собрано и размечено более 200 заголовков и новостных статей на различных языках.

5.4. Модуль Event Extraction

Для реализации модуля извлечения информации используется инструмент Томита-парсер [20]. Извлечение данных осуществляется с помощью шаблонов (контекстно - свободных грамматик, которые называются КС-грамматиками) и с использованием словарей ключевых слов. Парсер решает необходимые задачи обработки текста: токенизацию, сегментацию,

морфологический анализ. В системе отсутствует визуальная среда разработки, управление осуществляется с помощью командной строки. Пример представлен на рисунке 13.

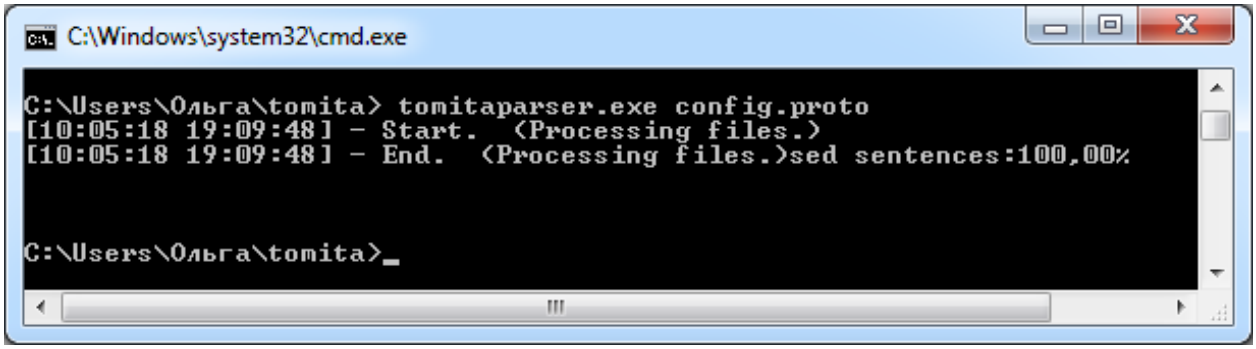

Рис. 13: Запуск Томита-парсера.

Событие – тройка <Subject, Predicate, Object>. Каждое из полей может содержать в себе несколько слов. Для одного события может быть найдено несколько полей. Так как основная функция заголовка состоит в том, чтобы привлечь внимание, ключевая информация находится в начале заголовка. Зачастую новостной заголовок строится следующим образом: <Субъект Предикат Объект Остальные слова> или <Объект Предикат Субъект Остальные слова>. Эта особенность учитывается при построении модуля извлечения события.

На вход программе подается txt файл заголовков русскоязычных новостей. Пример собранных новостных заголовков на русском языке представлен на рисунке 14.

```
1  Трамп пригласил Путина в Белый дом.
2  Турция назвала виновного в убийстве российского посла.
3  Сирийские ополченцы обстреляли американскую базу под Раккой.
4  Россиянам пообещали задержки с американскими визами.
5  Генконсульство России в Сиэтле отчиталось о своем закрытии.
6  Летчик Ярошенко рассказал о новых издевательствах в американской тюрьме.
7  Трамп обвинил The Washington Post в поддержке Amazon.
8  С американского генконсульства в Петербурге сняли флаг.
9  Россия вышлет 50 британских дипломатов.
10 США обвинили Иран в дестабилизации Ближнего Востока.
11 Израильские солдаты расстреляли палестинцев в секторе Газа.
12 Россия решила выслать европейских дипломатов.
13 Россия усилила военную мощь в Сирии.
14 Чехия выдала российского хакера Никулина США.
15 Папа Римский отверг существование ада.
16 США решили выслать из страны торгового представителя России.
17 Россия вышлет 60 американских дипломатов.
18 В Венесуэле заключенные подожгли тюрьму для побега и сгорели.
19 МИД предложил закрыть консульство США в Петербурге.
20 Ассанжа лишили связи с внешним миром.
```

Рис. 14: Заголовки русскоязычных новостей.

Predicate

Под предикатом будем понимать «действие». В качестве предиката будем извлекать подцепочки следующего вида:

1. Глагол (в личной форме).
   Пример: «объявили».
2. Глагол (в личной форме) + Глагол (инфинитив).
   Пример: «отказался расследовать».
3. Краткое причастие.
   Пример: «построен» из цепочки «дом построен».
4. Любой из пунктов 1-2 + пункт 3.
   Пример: «был построен» из цепочки «дом был построен».
5. Частица «не» + любой из пунктов 1-4.
   Пример: «не поддержит» из цепочки «руководство не поддержит это решение». Необходимо включать частицу «не» в рассмотрение, иначе суть действия будет извлечена неверно. Из выше приведенного примера, в случае отсутствия пункта 5 извлеклось бы только слово «поддержит».

Subject

Под субъектом будем понимать «того, кто совершает действие». Под словом «группа» будем подразумевать цепочку подряд идущих слов, удовлетворяющих указанному условию. В качестве субъекта будем извлекать подцепочки следующего вида:

1. Существительное (в именительном падеже)
   Пример: «день» из цепочки «хороший день».
2. Группа из пункта 1.
   Пример: «повар-кондитер».
3. Слово с большой буквы, которое содержится в словаре и является именем, фамилией или отчеством.
   Пример: «Дмитрий», «Иванов».
4. Группа из пункта 3.
   Пример: «Дмитрий Иванов».
5. Прилагательное (геополитическое) + цепочка из пунктов 1-2 – согласованные между собой по роду, числу и падежу.
   Пример: «хорватский гроссмейстер». Отметим, что в случае идентификации международного события, если существует геополитическое прилагательное, которое относится к существительному, то важно его

извлечь. Например, при извлечении субъекта из цепочки «российский президент» необходимо извлечь цепочку полностью, так как в противном случае теряется важный аспект события.

6. Существительное, которое содержится в географическом словаре и является аббревиатурой.
   Пример: «РФ».
7. Существительное, которое является аббревиатурой.
   Пример: «ТРЦ».
8. Пункт 5 + пункт 7.
   Пример: «российский МИД».
9. Слово с большой буквы, которое написано латиницей.
   Пример: «Youtube».
10. Любой из пунктов 1-2 + любой из пунктов 3-4.
    Пример: «Поезд Ким Чен Ына».
11. Любой из пунктов 1-2 + существительное, которое находится в географическом словаре.
    Пример: «Канцлер Австрии».

Object

Под объектом будем понимать «того, по отношению к кому совершают действие». Таким образом, извлечение объектов из текста аналогично задаче извлечения субъекта, без привязки к именительному падежу.

Для работы парсера требуются следующие исходные файлы: config.proto – конфигурационный файл, dic.gzt – корневой словарь, grammar.cxx – грамматика, fact_types.proto – файл описания типов фактов, kw_types.proto – файл описания типов ключевых слов.

Файл grammar.cxx содержит в себе правила, написанные на языке контекстно-свободных грамматик. В качестве примера на рисунке 15 приведены правила для извлечения Predicate из текста. Исходный код всех файлов модуля представлен в Приложении 4.

```
1  Predicate -> Verb<rt>;
2  Predicate -> Verb<rt> Word<gram="V">;
3  Predicate -> Word<gram="V, кр">;
4  Predicate -> Verb<rt> Word<gram="V, кр"> | Verb<rt> Word<gram="V"> Word<gram="V, кр">;
5  Predicate -> ('не') Verb<rt> | ('не') Verb<rt> Word<gram="V"> | ('не') Word<gram="V, кр">|
6               ('не') Verb<rt> Word<gram="V, кр"> | ('не') Verb<rt> Word<gram="V"> Word<gram="V, кр">;
```

Рис. 15: Правила для извлечения Predicate из текста.

На рисунке 16 представлен отладочный HTML файл работы модуля для предложения – заголовка.

Высокопоставленные российские силовики попали под американские санкции .

Event		
Predicate	**Subject**	**Object**
попали	российские силовики	американские санкции

Рис. 16: Отладочный файл работы модуля Event Extraction.

5.5. Модуль Tomita Results Post Processing

Модуль состоит из двух компонент: translating и synonyms searching, т.е. перевод и поиск синонимов. В результате работы предыдущего модуля имеется список новостных заголовков и извлеченные из него тройки <Subject, Predicate, Object>, которые могут состоять из нескольких полей, внутри которых находятся ключевые слова. Для последующего поиска ключевых слов в альтернативной ленте их необходимо перевести.

Translating

Перевод ключевых слов осуществляется с помощью API Яндекс Переводчика [30]. Клиент отправляет http-запрос на сервер, сервер возвращает ответное сообщение клиенту. HTTP – протокол запроса-ответа в модели клиент-сервер. Для осуществления превода, в теле HTTP запроса к API необходимо указать текст, который надо перевести, язык с которого осуществляется перевод, в нашем случае это всегда русский язык, и язык назначения перевода. Примеры отправляемого запроса и получаемого ответа представлены на рисунках 17 и 18. Исходный код модуля представлен в Приложении 5.

```
POST /api/v1.5/tr.json/translate?lang=en-ru&key=API-KEY HTTP/1.1
Host: translate.yandex.net
Accept: */*
Content-Length: 17
Content-Type: application/x-www-form-urlencoded

text=Hello World!
```

Рис. 17: Пример запроса.

Synonyms searching

Для улучшения эффективности поиска ключевых слов в альтернативной ленте, а также для решения задачи кореференции, производится поиск их синонимов. Модуль реализуется с использованием публичного

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json; charset=utf-8
Content-Length: 68
Connection: keep-alive
Keep-Alive: timeout=120
X-Content-Type-Options: nosniff
Date: Thu, 31 Mar 2016 10:50:20 GMT
{
    "code": 200,
    "lang": "en-ru",
    "text": [
        "Здравствуй, Мир!"
    ]
}
```

Рис. 18: Пример ответа.

API, предоставляемого сервисом thesaurus.altervista [31]. Данный сервис позволяет искать синонимы для многих языков, на основе открытых словарей OpenOffice. Пример работы с данным API изображен на рисунке 19. Исходный код модуля представлен в Приложении 6.

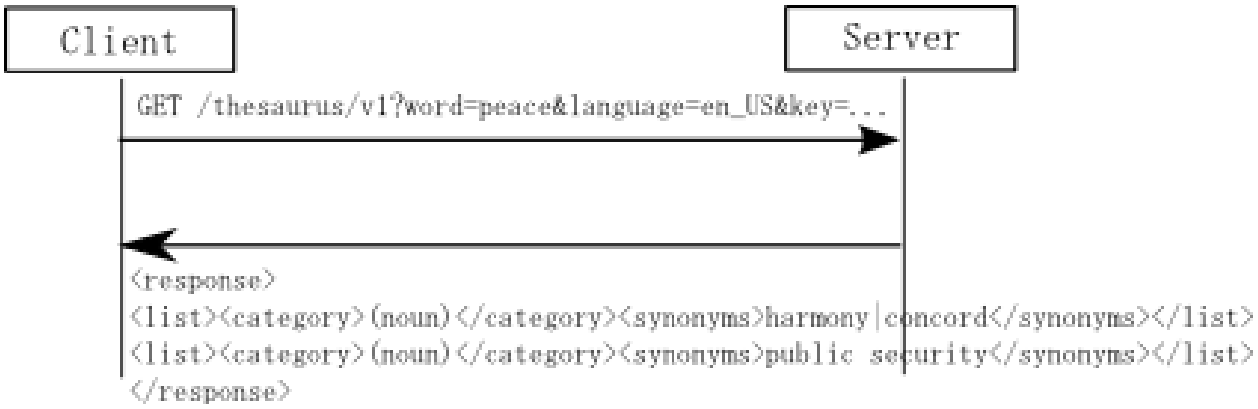

Рис. 19: Принцип взаимодействия с API thesaurus.altervista.

5.6. Модуль Lemmatisation

Для поиска ключевых слов в альтернативных источниках информации необходимо привести их к нормальному виду, т.е. произвести лемматизацию. Лемматизация – это переход от словоформы к лемме слова. Помимо ключевых слов, производится лемматизация новостных сообщений из альтернативной ленты.

Модуль лемматизации реализован на основе библиотек OpenSource проекта LemmaGenerator [32]. Данный сервис выполнен за счет хранения основных наборов лемм для различных языков и дальнейшего анализа требуемого текста на предмет их соответствия. В данной работе использовался поставляемый вместе с лемматизатором набор списков лемм. Пример использования лемматизатора LemmaGenerator представлен на рисунке 20. Исходный код модуля представлен в Приложении 7.

```
var path = "/path/to/the/lemmatizer/file"; // Путь до файла с леммами.
var stream = File.OpenRead(dataFilePath); // Создание потока чтения файла.
var lemmatizer = new Lemmatizer(stream); // Инициализация лемматизатора.
var result = lemmatizer.Lemmatize("words"); // Получение леммы словоформы.
```

Рис. 20: Пример использования LemmaGenerator.

5.7. Модуль World News Determination

В результате работы предыдущих модулей на вход модуля World News Determination подается два типа данных:

1. Тройки <Subject, Predicate, Object>. Каждый <Subject>, <Predicate>, <Object> может содержать в себе одно или несколько полей. <Subject> и <Object> могут не содержать полей (но не одновременно), наличие поля в <Predicate> является обязательным. Каждое из полей содержит в себе ключевые слова или слово. Указанные слова переведены на языки альтернативной новостной ленты (с учетом синонимичных слов) и нормализованы (произведена лемматизация).
2. Заголовки и содержания новостей альтернативной ленты. Указанные данные нормализованы.

Для определения международного события производится поиск ключевых слов в заголовках и новостных сообщениях альтернативной ленты. Осуществляется поиск одного из полей тройки <Subject, Predicate, Object> и двоек: <Subject, Predicate>, <Predicate, Object>, <Subject, Object>.

Пример выходного файла работы модуля представлен на рисунках 21, 22. Исходный код модуля представлен в Приложении 8.

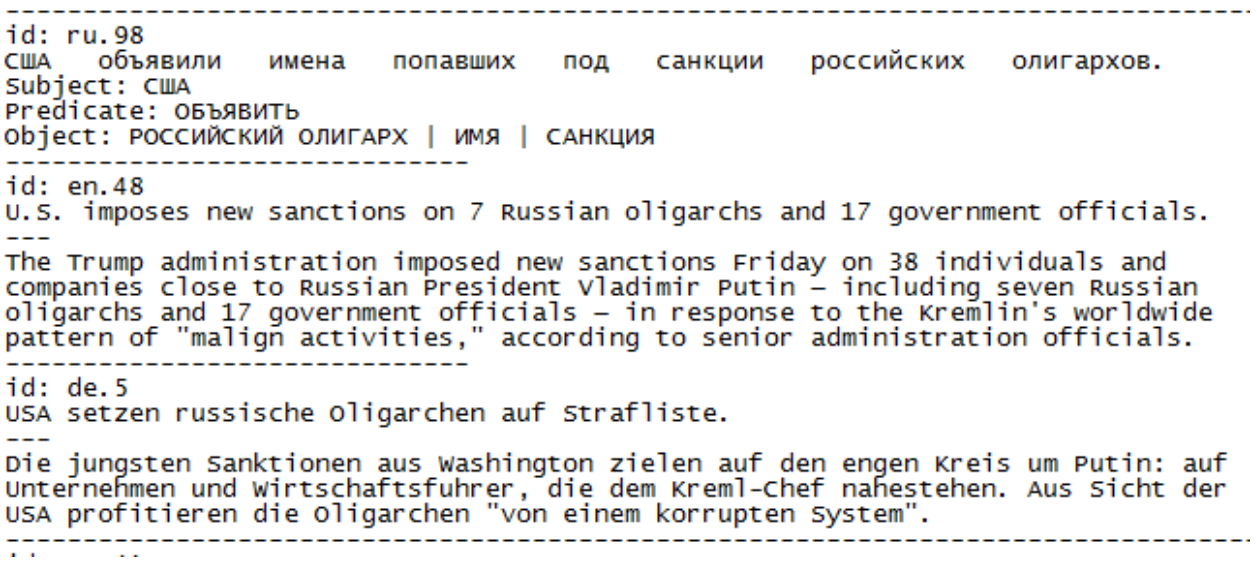

Рис. 21: Пример работы модуля World News Determination.

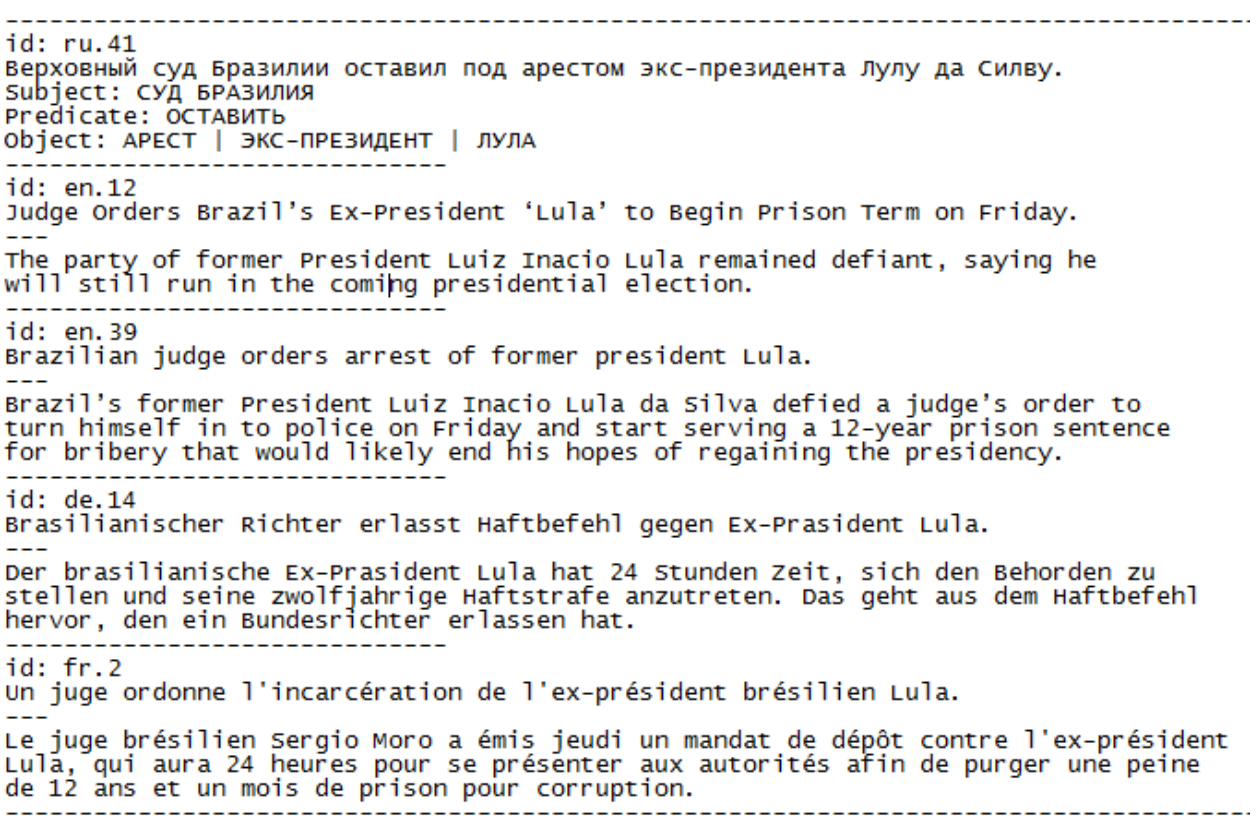

Рис. 22: Пример работы модуля World News Determination.

5.8. Оценка эффективности

В обучающей выборке содержится 234 документа. В результате ручной разметки было определено 110 событий основной ленты пользователя, из которых 33 являются международными.

С учетом определений, описанных в параграфе 1.6, имеем: $K = 33$, $C = 26$, $I = 4$. Таким образом, оценка точности и полноты, согласно фор-

муле (1.1) имеет следующий вид:

$$P = 0.867,\ R = 0.788$$

В качестве дополнительной оценки рассмотрим F-меру при $\beta = 1$. Таким образом, с учетом формулы (1.2) имеем оценку:

$$F = 0.825$$

Выводы

В данной работе продемонстрирована возможность построения автоматизированной системы определения международного события и реализован инструмент предоставления конечному пользователю информации об уровне значимости события.

В ходе работы были рассмотрены и изучены различные средства и подходы к интеллектуальному анализу текста для корректного извлечения события из русскоязычной новости.

Также была представлена реализация предложенной системы с использованием Томита-парсера и языка программирования C#. Полученная реализация полностью отвечает поставленным перед ней требованиям:

1. Сбор исходных данных.
2. Обработка новостных документов.
3. Определение и извлечение событий из новости.
4. Проверка события, для определения является ли событие международным.
5. Предоставление пользователю новости об идентифицированном событии из альтернативных источников.

Был произведен анализ эффективности работы построенной системы: точность и полнота равны 0.86 и 0.79 соответственно. Из чего можно сделать вывод о том, что данный подход имеет положительные результаты и решает поставленную задачу в достаточной мере для практического применения.

Заключение

В ходе работы построена автоматизированная система, позволяющая исследовать русскоязычные новости на предмет их масштаба. Показана эффективность предлагаемого подхода и его практическая применимость.

Заметим, что работа построена на исследовании русскоязычных новостей, но предложенный подход применим и для новостей на произвольном языке. Также, при исследовании новостей из некоторой конкретной области (например, спорт) можно достичь более высоких показателей точности и полноты.

Все вышеперечисленное может стать предметом дальнейших изучений.

Список литературы

[1] Барсегян А. Анализ данных и процессов. 3 изд. БХВ-Петербург, 2009. 512 с.

[2] Klügl P. Context-specific Consistencies in Information Extraction: Rule-based and Probabilistic Approaches. BoD – Books on Demand, 2015. 208 с.

[3] Piskorski J., Yangarber R. Information Extraction: Past, Present and Future. // Poibeau T., Saggion H., Piskorski J., Yangarber R. Multi-source, Multilingual Information Extraction and Summarization. Theory and Applications of Natural Language Processing. Springer, Berlin, Heidelberg, 2013.

[4] Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие. М.: Изд-во НИУ ВШЭ, 2017. 269 с.

[5] Извлечение объектов и фактов из текстов в Яндексе. Лекция для Малого ШАДа. https://habr.com/company/yandex/blog/205198/

[6] Wimalasuriya D., Dou D. Ontology-based information extraction: An introduction and a survey of current approaches // Journal of Information Science, 2010. Vol.36, No 3, P. 306-323.

[7] Википедия — свободная энциклопедия. https://ru.wikipedia.org/

[8] Siefkes C., Siniakov P. An Overview and Classification of Adaptive Approaches to Information Extraction. // Journal on Data Semantics IV, 2005. Vol. 3730, P. 172-212.

[9] SAIC Information Extraction. https://www-nlpir.nist.gov/related_projects/muc/

[10] Grishman R., Sundheim B. Message Understanding Conference-6: a brief history // Proceedings of COLING-1996, 1996. P. 466-471.

[11] Doddington G., Mitchell A., Przybocki M., Ramshaw L., Strassel S., Weischedel R. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. // Proceedings of the 7th Language Resources and Evaluation Conference, 2004.

[12] ACE. Linguistic Data Consortium. https://www.ldc.upenn.edu/collaborations/past-projects/ace

[13] Диалог. Конференция по компьютерной лингвистике. http://www.dialog-21.ru/

[14] General Architecture for Text Engineering https://gate.ac.uk/

[15] Apache OpenNLP. https://opennlp.apache.org/index.html

[16] DBpedia. http://wiki.dbpedia.org/

[17] The Stanford Natural Language Processing Group. https://nlp.stanford.edu/software/openie.html

[18] Stanford CoreNLP – Natural language software. https://stanfordnlp.github.io/CoreNLP/

[19] MAchine Learning for LanguagE Toolkit. http://mallet.cs.umass.edu/

[20] Томита-парсер – Технологии Яндекса. https://tech.yandex.ru/tomita/

[21] PullEnti - SDK извлечение именованных сущностей из неструктурированных текстов (Puller of Entities). http://www.pullenti.ru/

[22] DBpedia Version 2014 released. http://blog.dbpedia.org/2014/09/09/dbpedia-version-2014-released/

[23] Тьюринг А. Может ли машина мыслить? М.: ГИФМЛ, 1960. 110 с.

[24] Alan Turing. https://news.harvard.edu/gazette/story/2012/09/alan-turing-at-100/

[25] Jurafsky D., Martin J., Norvig P., Russell S. Speech and Language Processing. Second Edition. Pearson Education, 2014. 1024 с.

[26] Jones K.S. Natural Language Processing: A Historical Review. // Current Issues in Computational Linguistics: In Honour of Don Walker, 1994. Vol 9.

[27] IBM Watson. https://www.ibm.com/watson/

[28] Manning C., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008.

[29] Боярский К. К. Введение в компьютерную лингвистику: учеб. пособие. СПб: НИУ ИТМО, 2013. 72 с.

[30] API Переводчика — Технологии Яндекса. https://tech.yandex.ru/translate/

[31] Thesaurus. https://thesaurus.altervista.org/

[32] LemmaGenerator: Generator of rule-based lemmatizers (based on examples) for serveral European languages. https://github.com/AlexPoint/LemmaGenerator

Приложение 1

Список обозначений и сокращений:

Natural Language – естественный язык

Information Extraction, IE – извлечение информации

Natural Language Processing, NLP – обработка естественного языка

Text Mining, TM – интеллектуальный анализ текста

Machine Learning, ML – машинное обучение

Named Entity Recognition, NER – распознавание именованных сущностей

Coreference resolution, CO – разрешение кореференции

Information Retrieval, IR – информационный поиск

Rule-based – на основе правил

Ontology-based – на основе онтологий

Precision – точность

Recall – полнота

Part-of-speech tagging, POS-tagging – разметка по частям речи

Command line interface, CLI – интерфейс командной строки

Application programming interface, API – программный интерфейс приложения

Приложение 2

Список русскоязычных RSS-каналов:
https://lenta.ru/rss/news
https://lenta.ru/rss/news/world
http://tass.ru/rss/export/index.xml?feed=v2&sections=NDg5OA%3D%3D
https://www.vesti.ru/vesti.rss
https://www.kommersant.ru/RSS/section-world.xml
https://www.kommersant.ru/RSS/news.xml

Список англоязычных RSS-каналов:
https://www.theguardian.com/world/rss
http://feeds.bbci.co.uk/news/world/rss.xml
http://feeds.washingtonpost.com/rss/world
http://rss.nytimes.com/services/xml/rss/nyt/World.xml
http://rssfeeds.usatoday.com/UsatodaycomWorld-TopStories
http://www.independent.co.uk/news/world/rss

Список французскоязычных RSS-каналов:
http://www.lefigaro.fr/rss/figaro_international.xml
http://www.lemonde.fr/international/rss_full.xml
http://rss.liberation.fr/rss/10/

Список немецкоязычных RSS-каналов:
http://rss.dw.com/xml/rss-de-all
http://www.faz.net/rss/aktuell/politik/
https://www.welt.de/feeds/latest.rss

Приложение 3

Листинг модуля Input Data Aggregation. Язык: C#.

```
internal class Program
{
public static void Main(string[] args)
{
List<string> urls = GetRssFeeds();
const string fileName = "news.csv";

SyndicationFeed feed = new SyndicationFeed();
foreach (var url in urls)
{
try
{
var reader = XmlReader.Create(url);
feed.Items = feed.Items.Concat(SyndicationFeed.Load(reader)
.Items);
reader.Close();
}
catch (Exception ex)
{
Console.WriteLine($"Не могу прочитать новости с {url}");
}
}

var NewsList = feed.Items.Select(item => new Article(item))
.ToList();

using (var csv = new CsvWriter(new StreamWriter(fileName, false,
 encoding: Encoding.GetEncoding(1251))))
{
csv.Configuration.Delimiter = ";";
csv.Configuration.CultureInfo = CultureInfo
.DefaultThreadCurrentUICulture;
foreach (var article in NewsList)
{
csv.WriteField(article.Id);
csv.WriteField(article.Date);
csv.WriteField(article.Title);
```

```
csv.WriteField(article.Content?.Trim());
csv.WriteField(article.Tags);
csv.WriteField(article.Link);
csv.NextRecord();
}
}
Console.WriteLine("Done!");
Console.ReadKey();
}

private static List<string> GetRssFeeds()
{
List<string> feedLinks = new List<string>();
try
{
foreach (string line in File.ReadLines(@"feeds.txt"))
{
feedLinks.Add(line.Trim());
}
}
catch (Exception e)
{
Console.WriteLine("не могу прочесть файл");
Console.ReadKey();
}

return feedLinks;
}
}

public class Article
{
public string Id;
public DateTime Date;
public string Title;
public string Content;
public string Tags;
public string Link;

public Article(SyndicationItem item)
{
this.Id = Guid.NewGuid().ToString();
```

```
this.Date = item.PublishDate.DateTime;
this.Title = item.Title.Text?.AppendPoint();
this.Content = item.Summary?.Text?.AppendPoint();
this.Tags = string.Join(" | ", item.Categories.Select(el =>
 el.Name));
this.Link = item.Id;
}
}

public static class StringExtension
{
public static string AppendPoint(this String str)
{
if (str.Length > 0 && str.Last() != '.')
str += '.';
return str;
}
}
```

Приложение 4

Модуль Event Extraction. Исходные файлы проекта для Томита-парсера.

config.proto:

```
encoding "utf8";

TTextMinerConfig {
Dictionary = "dic.gzt";

Input = {File= "rus_news.txt" Type = dpl}

Output = {File = "output.xml" Format = xml}

Articles = [{Name = "извлечение_события_грамматика"}]

Facts = [{Name = "Event"}]

PrettyOutput = "pretty.html";
}
```

event_extraction_grammar.cxx:

```
#encoding "utf8"

Predicate -> (’не’) Verb<rt> ( Word<gram="V">+ ) |
(’не’) Word<gram="V, кр">+;
Event -> Predicate interp (Event.Predicate::not_norm);


N -> Noun<wff=~/[А-Я]{1}/,gram="им, ~гео, ~имя, ~фам, ~отч",
~quoted,rt>;

Persn -> Word<h-reg1, gram="имя", ~quoted> | Word<h-reg1,
 gram="фам", ~quoted>|Word<h-reg1, gram="отч", ~quoted>;
Persn -> Word<wff=/[А-Я]{1}[а-я]{2,9}/, gram="~V, ~A, ~PR,
~ADV, ~anim, ~inan", ~quoted>;
Name -> Persn+;

Loc -> Noun<gram="гео, сокр">;
```

```
Loc -> Noun<gram="гео, им, ~род, ~дат">;

GeoAdjNoun -> Adj<kwtype=["geoadj"], gnc-agr[1]> N<gnc-agr[1],rt>;

Abbr -> Noun<wff=/[А-Я]{2,6}/, gram="сокр">;

CompanyLat -> Word<h-reg1, lat>+;

NCompany -> N Word<h-reg1, quoted>;
NName -> N<rt> Name;
NLoc -> N<rt> Noun<gram="гео">;

Subject -> N|Name|Loc|GeoAdjNoun|NName|NLoc|Abbr
|CompanyLat|NCompany;

Phrase -> Prep Noun|Adv|Adj<kwtype=~["geoadj"], gnc-agr[1]>
Noun<gnc-agr[1]>;

Event -> Subject<sp-agr[1]> interp (Event.Subject::not_norm)
(Phrase+)
Predicate<sp-agr[1]> interp (Event.Predicate::not_norm);


ObN -> Noun<wff=~/[А-Я]{1}/>;
ObGeoAdjNoun -> (Adj<kwtype=["geoadj"], gnc-agr[1]>)
ObN<gnc-agr[1]>;
Object -> ObN|ObGeoAdjNoun|Name;

Event -> Object interp (Event.Object::not_norm);
```

dic.gzt:

```
encoding "utf8";
import "base.proto";
import "articles_base.proto";
import "fact_types.proto";
import "kw_types.proto";
import "exception.gzt";
import "geoadj.gzt";

TAuxDicArticle "извлечение_события_грамматика"
{
```

```
key={"tomita:event_extraction_grammar.cxx" type=CUSTOM}
}
```

fact_types.proto:

```
import "base.proto";
import "facttypes_base.proto";

message Event: NFactType.TFact
{
optional string Predicate = 1;
optional string Subject = 2;
optional string Object = 3;
}
```

kw_types.proto:

```
import "base.proto";
import "articles_base.proto";
import "kwtypes_base.proto";

message geoadj : TAuxDicArticle {}
```

Приложение 5

Листинг модуля Tomita Results Post Processing: Translating. Язык: C#.

```
 public class Translator
{
private IYandexTranslator translator = Yandex.Translator
.Yandex.Translator(api => api.ApiKey("API-KEY").Format
(ApiDataFormat.Json));

private string language;
public Translator(string language)
{
if (language == "en")
this.language = "ru-en";
if (language == "de")
this.language = "ru-de";
if (language == "fr")
this.language = "ru-fr";
}

public void Translate(List<Event> evnts)
{
List<Word> toTranslate = new List<Word>();
foreach (var evnt in evnts)
{
if (evnt.Objects != null)
toTranslate.AddRange(evnt.Objects);
if (evnt.Subjects != null)
toTranslate.AddRange(evnt.Subjects);
if (evnt.Predicates != null)
toTranslate.AddRange(evnt.Predicates);
}

string neddTranslate = string.Join(". ",
 toTranslate.Select(el => el.OriginalText));
string translated = translator.Translate(language,
 neddTranslate).Text;

var translatedWords = translated.Split('.').Select(el =>
```

```
el.Trim().RemoveToInBeginning()).ToArray();
for (int i = 0; i < toTranslate.Count; i++)
{
toTranslate[i].TranslatedText = translatedWords[i];
}
}
}
```

Приложение 6

Листинг модуля Tomita Results Post Processing: Synonyms searching. Язык: C#.

```
 public class SynSearcher
{
private string language;
private Dictionary<Tuple<string, string>,
List<string>> CachedResults = new Dictionary
<Tuple<string, string>, List<string>>();
public SynSearcher(string lang)
{
if (lang == "en")
language = "en_US";
if (lang == "de")
language = "de_DE";
if (lang == "fr")
language = "fr_FR";
}

public void PopulateSynonims(List<Event> events)
{
var wordsList = events.SelectMany(el => el.Objects).ToList();
wordsList.AddRange(events.SelectMany(el => el.Subjects));
wordsList.AddRange(events.SelectMany(el => el.Predicates));

foreach (var word in wordsList)
{
if (!CachedResults.ContainsKey(Tuple.Create(word.TranslatedText,
 word.PartOfSpeech)))
{
word.Synonyms = GetSynonyms(word);
CachedResults[Tuple.Create(word.TranslatedText,
 word.PartOfSpeech)] = word.Synonyms;
}
else { word.Synonyms = CachedResults[Tuple
.Create(word.TranslatedText, word.PartOfSpeech)]; }
}
}
```

```
private List<string> GetSynonyms(Word word)
{
string url = "http://thesaurus.altervista.org/thesaurus/v1";
string serverAddress = url + "?word=" +
word.TranslatedText + "&language=" + language + "&key=" +
 "API-KEY" + "&output=" + "json";
HttpWebRequest request = (HttpWebRequest)WebRequest
.Create(serverAddress);
try
{
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
if (response.StatusCode == HttpStatusCode.OK)
{
string myResponse = "";
using (System.IO.StreamReader sr = new System.IO.StreamReader
(response.GetResponseStream()))
{
myResponse = sr.ReadToEnd();
}
return JsonConvert.DeserializeObject<RootObject>(myResponse)
.ParseSynonyms(word.PartOfSpeech);
}
}
catch
{
Console.WriteLine($"No synonyms for {word.TranslatedText}
 found", Color.Red);
return new List<string>();
}

return new List<string>();
}
}

public class List
{
public string category { get; set; }
public string synonyms { get; set; }
}

public class Response
{
```

```
public List list { get; set; }
}

public class RootObject
{
public List<Response> response { get; set; }

public List<string> ParseSynonyms(string pos)
{
var result = response.Where(el => el.list.category.Contains(pos))
.SelectMany(el => el.list.synonyms.Split('|')).Where(el => el
.Contains("(generic term)") || !el.Contains("("))
.Select(el => (el.IndexOf("(generic term)") > 0
 ? el.Remove(el.IndexOf("(generic term)")) : el).Trim()).Take(10)
 .ToList();
return result;
}
}
```

Приложение 7

Листинг модуля Lemmatisation. Язык: C#.

```
public class TextLemmatiser
{
private const string path = @"C:\TomitaXmlProcessor\Lemmatizer\";
private Lemmatizer lemmatizer;
public TextLemmatiser(string language)
{
string filePath = "";
if (language == "en")
filePath = path + @"full7z-mlteast-en-modified.lem";
if (language == "de")
filePath = path + @"full7z-multext-ge.lem";
if (language == "fr")
filePath = path + @"full7z-mlteast-fr.lem";

using (var stream = File.OpenRead(filePath))
{
lemmatizer = new Lemmatizer(stream);
}
}

private void LemmatizeArticle(Article article)
{
var text = article.Title + article.Content;
var punctuation = text.Where(Char.IsPunctuation)
.Distinct().ToArray();
var words = text.Split().Select(x =>
x.Trim(punctuation)).ToArray();
var result = "";
foreach (var word in words)
{
result += lemmatizer.Lemmatize(word) + " ";
}
article.StemmedText = result;
}

private void LemmatizeWord(Word word)
{
```

```
word.LemmatizedText = lemmatizer.Lemmatize(word.TranslatedText);
foreach (var synonym in word.Synonyms)
{
word.LemmatizedSynonyms.Add(lemmatizer.Lemmatize(synonym));
}
}

public void LemmatizeArticles(IEnumerable<Article> articles)
{
foreach (var article in articles)
{
LemmatizeArticle(article);
}
}

public void LemmatizeWords(IEnumerable<Word> words)
{
foreach (var word in words)
{
LemmatizeWord(word);
}
}
}
```

Приложение 8

Листинг модуля World News Determination. Язык: C#.

```
public class EventsInNewsSearcher
{
private List<Article> articles;
private List<Event> events;

public EventsInNewsSearcher(List<Article> articles,
 List<Event> events)
{
this.articles = articles;
this.events = events;
}

public List<Tuple<Article, Event>> FindArticlesByEvents()
{
List<Tuple<Article, Event>> result = new List
<Tuple<Article, Event>>();
foreach (var evnt in events)
{
foreach (var article in articles)
{
if (IsArticleContainsEvent(article, evnt))
{
result.Add(Tuple.Create(article, evnt));
break;
}
}
}
return result;
}

private bool IsArticleContainsEvent(Article article, Event evnt)
{
int isContainsObject = 0;
int isContainsSubject = 0;
int isContainsPredicate = 0;
foreach (var word in evnt.Objects)
{
```

```
foreach (var singleWord in word.LemmatizedSynonyms
.Union(new[] { word.LemmatizedText }))
if ((article.StemmedText /*+ article.Content*/)
.ToLower().Contains(singleWord.ToLower()))
{
isContainsObject += 1;
break;
}
}
foreach (var word in evnt.Subjects)
{
foreach (var singleWord in word.LemmatizedSynonyms
.Union(new[] { word.LemmatizedText }))
if ((article.StemmedText /*+ article.Content*/)
.ToLower().Contains(singleWord.ToLower()))
{
isContainsSubject += 1;
break;
}
}
foreach (var word in evnt.Predicates)
{
foreach (var singleWord in word.LemmatizedSynonyms
.Union(new[] { word.LemmatizedText }))
if ((article.StemmedText /*+ article.Content*/)
.ToLower().Contains(singleWord.ToLower()))
{
isContainsPredicate += 1;
break;
}
}
if (isContainsObject + isContainsSubject + isContainsPredicate > 2)
{
return true;
}
return false;
}
}
```